ISSN 1392-1215 ELEKTRONIKA IR ELEKTROTECHNIKA. 2005. Nr. 7(63)

T 170 ELEKTRONIKA

Juodosios dėžės modelių testų kūrimas

V. Jusas, K. Paulikas, R. Šeinauskas

Programų inžinerijos katedra, Kauno technologijos universitetas,
Studentų g. 50, LT-51368 Kaunas, Lietuva; tel. +370 37 300609; el. p.: vacys@soften.ktu.lt; kestas@elen.ktu.lt; rimantas.seinauskas@ktu.lt

Įžanga

Mikroschemos projektuojamos nuosekliais etapais: specifikacijos rengimas, aukšto abstrakcijos lygio modelio sudarymas ir elgsenos modeliavimas, perėjimas prie žemesnio abstrakcijos lygio modelių (RTL, loginės schemos, analoginės schemos) ir tikslesnių būsimos schemos parametrų modeliavimas, topologijos braižymas, siuntimas į gamybą, gautų rezultatų testavimas. Viso šio proceso tikslas – gauti tinkamai veikiantį galutinį produktą.

Kiekvieno projektavimo žingsnio metu siekiama aptikti ir ištaisyti neišvengiamai pasitaikančias projektavimo klaidas. Klaidoms aptikti taikomi įvairūs metodai [1]. Vienais atvejais analizuojamas schemos veikimas, kitais – struktūra. Struktūros analizavimo metodai, tarp jų ir formalūs [2], turi nemažai privalumų, tačiau sėkmingam jų taikymui reikalingas ir atitinkamas projektuotojų pasirengimas bei programinė įranga.

Schemos veikimą analizuojantys modeliavimo metodai yra gerokai paprastesni, be to, jie leidžia aiškiau suvokti, kaip veiks būsimoji schema. Nesvarbu, su kokiais schemos modeliais dirbama, modeliavimui reikalingi įėjimo duomenys, kurie gali būti kuriami rankomis, automatiškai ir pusiau automatiškai bei atsitiktinai generuojami. Automatiniais testų kūrimo metodais (ATPG) stengiamasi sudaryti testą remiantis žiniomis apie schemą [3] (pvz., pagal loginės schemos struktūrą). Šiuo metu pramonėje taikomi ATPG metodai remiasi loginės schemos analize.

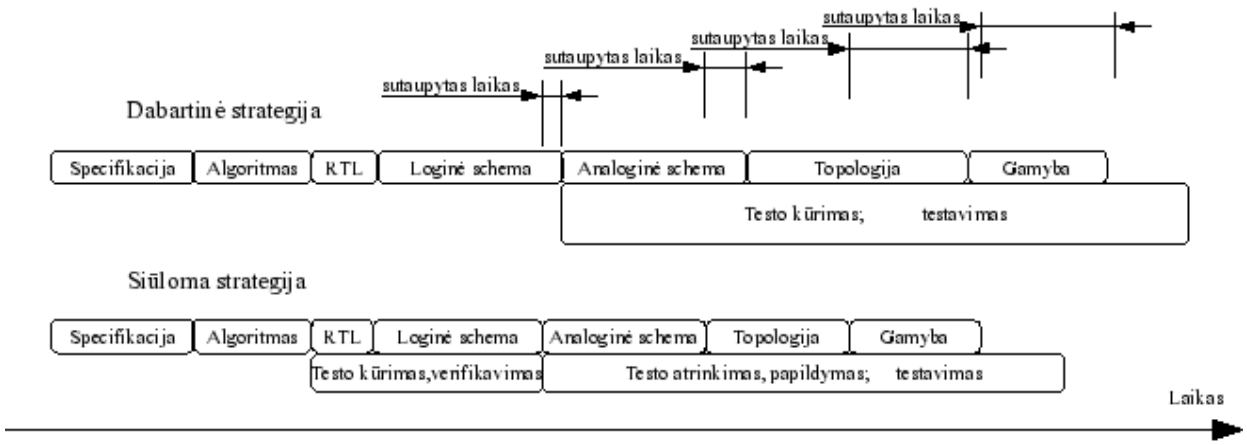
Testuojant aukšto abstrakcijos lygio modelius populiariausias sprendimas – atsitiktinai gaunami rinkiniai [4],[5]. Tačiau dėl didelės apimties tokie testai sunkiai pritaikomi vėlesniuose etapuose. Šiame straipsnyje aprašomi schemos elgsenos modeliavimu pagrįsti metodai, leidžiantys sukurti gerokai trumpesnius nei atsitiktinis testus, išlaikant labai panašią jų kokybę.

Būsimos schemos testavimą tikslinga pradėti kuo ankstyvesniame schemos projektavimo etape, nes kuo vėliau klaida aptinkama, tuo brangiau ir tuo daugiau reikia laiko ją ištaisyti. Todėl šiame straipsnyje siūloma testų sudarymo strategija, leidžianti sudaryti testus, kai tik tampa prieinama galimybė modeliuoti būsimos schemos elgseną (1 pav.). Pagal šią strategiją sudaryti testai gali būti naudojami ir vėlesniuose projektavimo etapuose. Šie testai nepriklauso nuo konkrečios loginės schemos realizacijos. Be to, testinė seka vėliau gali būti papildoma atsižvelgiant į realizacijos detales.

Testo papildymas yra gerokai paprastesnis uždavinys nei testo sudarymas, todėl tai dar viena priežastis pradėti testų sudarymą ankstyvuose projekto etapuose.

Juodos dėžės modeliai

Kalbant apie juodosios dėžės modelius, turima galvoje, kad vartotojui nežinomos naudojamo posistemio realizacijos detalės. Pavyzdžiui, pradiniuose projekto



1 pav. Testavimo reikšmė projektuojant elektronines schemas

etapuose gali būti naudojamas prototipas, leidžiantis modeliuoti būsimos sistemos veikimą, tačiau neturintis jokios informacijos apie fizinę struktūrą ir parametrus. Taip pat jei naudojami kitų gamintojų komponentai, jų realizacijos detalės gali būti slepiamos siekiant apsaugoti intelektinės nuosavybės teises (pvz., RAMBUS).

Šie modeliai pranašesni tuo, kad jie prieinami inžinieriams beveik nuo pat projekto pradžios. Juodosios dėžės modeliai yra žymiai paprastesni nei žemesnio abstrakcijos lygio modeliai, todėl jų modeliavimas yra kur kas pigesnis tiek laiko, tiek reikalavimų aparatūrai požiūriu. Taip pat nėra didelių apribojimų, kokia kalba jie aprašomi (C/C++, SystemC, VHDL, Verilog ir t. t.).

Perdavimų matrica

Perdavimų matrica [6] – tai juodosios dėžės testų vertinimo kriterijus. Laikoma, kad tarp schemos įėjimų ir išėjimų yra elektriniai keliai, taigi galima įsivaizduoti $n \times m$ matricą, kurioje *n* – įėjimų, o *m* – išėjimų skaičius, o kiekvienas langelis atitinka perdavimą iš konkretaus įėjimo į konkretų išėjimą.

Atsižvelgiant į tai, kad kiekvienas kelias nuo įėjimo į išėjimą gali eiti per lyginį ar nelyginį inversijų skaičių, gaunama $2 \cdot n \times 2 \cdot m$ matrica, kurioje galima pažymėti visus šiuos kelius.

Šis kriterijus negarantuoja, kad bus surasti visi galimi elektriniai keliai nuo kiekvieno įėjimo į išėjimą jau vien dėl to, kad pagal abstraktų modelį gali būti suprojektuotos kelios skirtingos realizacijos.

Šiam trūkumui sušvelninti galima modifikuoti perdavimų matricą, kad kiekvienas galimas kelias būtų pereinamas daugiau nei vieną kartą, arba analizuoti įėjimų poras (3D matrica).

Didžiausias šio kriterijaus privalumas – tiesinis algoritmo sudėtingumas ir paprastumas. Toliau aprašomi testų sudarymo metodai remiasi perdavimų matricos pildymu.

Testinė seka

Projekto vykdymo metu įvairaus abstrakcijos lygio modeliuose daromi pakeitimai siekiant ištaisyti pastebėtas klaidas, pagerinti charakteristikas ir t. t. Po kiekvieno tokio pakeitimo būtina įsitikinti, kad neįvyko nepageidaujamų schemos elgsenos pokyčių ir kad žemesnio abstrakcijos lygio modeliuose padaryti atitinkami pakeitimai. Taigi akivaizdu, kad kuo vėliau aptinkama klaida aukšto abstrakcijos lygio modeliuose, tuo brangiau kainuoja ją ištaisyti.

Mažoms kombinacinėms schemoms (apie 20 įėjimų) testuoti galima naudoti išsamų testą, kurį sudaro visos galimos įėjimų kombinacijos. Didesnėms schemoms išsamus testas sunkiai pritaikomas dėl jo ilgio (2^n , *n* – įėjimų skaičius), todėl naudojamos atsitiktinai kuriamos sekos. Jų ilgis parenkamas pagal norimą pasiekti testo kokybę.

Pasinaudojus juodosios dėžės modeliu ir perdavimų matrica, galima atsitiktinai sugeneruotą seką gerokai sutrumpinti beveik nepabloginant testo kokybės. Be to, šiuo būdu kuriant testą, gaunama papildomos informacijos apie schemą, kuri gali būti panaudota testinei sekai kryptingai papildyti.

Rinkinių atrinkimas

Turint testinę seką ir juodosios dėžės modelį, galima užpildyti perdavimų matricą ir sudaryti naują seką, į kurią bus įtraukti tik tie rinkiniai, kurie papildo perdavimų matricą. Šitaip sudaryto testo ilgio viršutinė riba yra $k \cdot (2 \cdot n \times 2 \cdot m)$, kur *n* – įėjimų, *m* – išėjimų skaičius, *k* – kiek kartų minimaliai pereinama vienu keliu iš įėjimo į išėjimą.

Buvo naudojamas testinių rinkinių atrinkimo pagal perdavimo matricos kriterijų algoritmas, kai vienu keliu iš įėjimo į išėjimą einama bent vieną kartą.

Algoritmui po vieną pateikiami rinkiniai iš atsitiktinai generuojamos ar kitaip gautos aibės. Paeiliui kiekvieno įėjimo reikšmė invertuojama, gautas naujas rinkinys *bv* modeliuojamas. Toliau tikrinama, kurių išėjimų reikšmės pasikeitė, ir pažymimas atitinkamas langelis perdavimų matricoje *M*, bei nustatomas požymis *papildė*. Rinkiniai, kuriems nustatytas požymis *papildė*, įtraukiami į testinę seką. Algoritmo sudėtingumas tiesinis.

Pažymėtina, kad galima atrinkti perteklinių rinkinių, t. y. tokių, kuriuos išmetus iš galutinės sekos matricos užpildymas nepasikeičia. Tai atsitinka todėl, kad vėliau atrinktas rinkinys gali užkloti visus anksčiau atrinkto rinkinio langelius. Praktikoje tokių rinkinių gaunama nedaug, o jų paieška padidintų algoritmo sudėtingumą. Jei svarbu pašalinti tokius rinkinius, galima atrinktą seką dar kartą įvykdyti tą patį algoritmą, pateikiant jam atrinktus rinkinius pradedant nuo paskutinio atrinkto.

Rinkinių generavimas

Nesunku pastebėti, kad perdavimų matricoje langeliai užpildomi poromis, tai yra jeigu užpildytas langelis, atitinkantis perdavimą $x_i \to y_j$, tai visada bus galimas ir perdavimas $\overline{x_i} \to \overline{y_j}$, todėl kiekvienoje keturių langelių grupėje, atitinkančioje perdavimus iš vieno įėjimo į vieną išėjimą, turi būti užpildytas lyginis langelių skaičius, be to, turi būti užpildyti gretimi pagal įstrižainę langeliai.

Pasinaudojant šia savybe, atrinktą testinę seką galima papildyti sudarant rinkinius, pildančius trūkstamus langelius. Kadangi žinomas bent vienas rinkinys užpildęs gretimą langelį, tai trūkstamas langelis gali būti užpildytas invertavus mus dominantį įėjimą rinkinyje. Esant didesnėms schemoms, šitaip galima gerokai padidinti perdavimų matricos užpildymą. Šis rinkinių papildymo būdas naudingiausias, kai taikomas pabaigus atsitiktinį kūrimą, kad būtų galima maksimaliai panaudoti atsitiktinio kūrimo galimybes (laikoma, kad naudingiau gauti kuo įvairesnius rinkinius, o sukuriamas rinkinys gaunamas labai panašus į jau atrinktą).

Euristiniu būdu buvo sudaryti keli algoritmai, kurie panaudojant perdavimų matricos duomenis, papildytų atsitiktinai kuriamą seką naujais rinkiniais. Ši idėja remiasi prielaida, kad tikslinga atidžiau išnagrinėti kiekvieno atrinkto rinkinio aplinką, nes atsitiktinai gauti panašius rinkinius gana sunku, ypač didelėms schemoms.

Visi testinės sekos papildymo algoritmai remiasi perdavimų matricos pildymo metu gaunama informacija. Šiai informacijai surinkti reikia pataisyti rinkinių atrinkimo algoritmą, kad būtų įsimenama informacija, kurie analizuojamo rinkinio įėjimai yra aktyvūs, t. y. jų reikšmės pakeitimas priešinga matomas išėjimuose.

Du papildymo algoritmai (*akt* ir *neakt*) kiekvienam atrinktam rinkiniui įterpia į atsitiktinę seką po vieną papildomą rinkinį, atitinkamai su invertuotais visais aktyviais ir visais neaktyviais įėjimais.

Dar vienas atsitiktinės sekos papildymo algoritmo variantas – kai vienam atrinktam rinkiniui sukuriama visa aibė rinkinių, kurie nuo originalo skiriasi tik vienu aktyviu įėjimu.

Naujai sukurti rinkiniai dedami i dėklą arba eilę ir vėliau po vieną pateikiami rinkinių atrinkimo algoritmui.

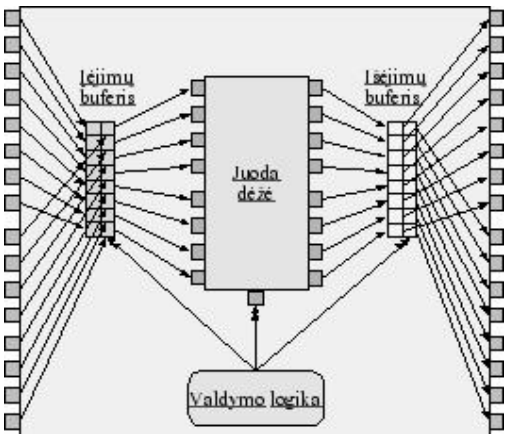
Schemos su atmintimi

Testų sudarymas schemoms su atmintimi, skirtingai nei kombinacinėms schemoms, vis dar yra sudėtingas uždavinys. Anksčiau aprašyti metodai gali būti taikomi schemų su atmintimi testams sudaryti, pritaikius iteracinį modelį.

Iteraciniame modelyje (2 pav.) schemai pateikiama visa seka rinkinių ir įsimenami išėjimai. Taip gaunamas kombinacinis schemos atitikmuo, t. y. galima įsivaizduoti, kad turime didelę kombinacinę schemą:

$$įėjimų \cdot kopijų \times išejimų \cdot kopijų$$

Kiekvienos rinkinių sekos pradžioje schema turi būti nustatoma į pradinę padėti; tam galima naudoti papildomą logiką arba paprasčiausiai kiekvieną rinkinių seką pradėti rinkiniais, nustatančiais schemą į pradinę padėtį. Valdymo logika siunčia sinchronizacijos signalus bei atitinkamus įėjimo signalus į schemą ir nustato, kur įsimenami išėjimai.



2 pav. Iteracinis modelis

Kopijų skaičiaus parinkimas tebėra tyrimų objektas. Schemą su atmintimi galima laikyti baigtiniu automatu, todėl kopijų skaičių stengiamasi parinkti tokį, kad jo užtektų visoms galimoms baigtinio automato būsenoms pasiekti.

Praktikoje šį skaičių nustatyti gali palengvinti grafiniai perdavimų matricos atvaizdai arba modelio aprašo teksto analizė, jeigu turime tokį tekstą.

Pabaigos sąlyga

Atrinkdami rinkinius pagal perdavimų matricą, turėtume baigti algoritmo darbą, kai surandame pakankamai daug rinkinių, kurie patikrintų visus kelius nuo įėjimo į išėjimą. Tačiau ši informacija nėra žinoma, t. y. keliai nuo įėjimo į išėjimą aptinkami algoritmo darbo metu ir, kol nepatikrintos visos 2^n įėjimo kombinacijos, negalime garantuoti, kad suradome visus kelius. Todėl reikalinga pabaigos sąlyga [7], kuri leistų užbaigti algoritmo darbą anksčiau.

Pabaigos sąlyga gali būti kuriamų rinkinių skaičius, algoritmo darbo trukmė, matematinė formulė, įvertinanti pasiektus rezultatus, ir pan. Eksperimente buvo naudojama euristiniu būdu suformuota matematinė sąlyga:

$$newlmt = c \cdot \sqrt{\Delta it^2 + \Delta matr^2} \ .$$

Čia c – laisvai parenkama konstanta (eksperimente buvo naudojama *0.5*); Δit – sukurtų rinkinių (iteracijų) skaičius imant nuo paskutinio atrinkto rinkinio; $\Delta matr$ – kiek langelių naujas rinkinys papildo matricą. Ši sąlyga įvertina tai, kad pradžioje rinkinių skaičius ir matricos užpildymas staigiai didėja (atitinkamai Δit mažas, o $\Delta matr$ didelis), o pabaigoje atvirkščiai. Buvo pastebėta, kad, pasinaudojus šia sąlyga, galima anksti nutraukti neefektyvius atsitiktinės sekos papildymo algoritmus.

Eksperimentas

Eksperimentuose buvo naudojamos ITC'99 etaloninio rinkinio schemos. Šį rinkinį sudaro schemos su atmintimi. Schemų VHDL elgsenos modeliai buvo perrašyti C kalba, norint paspartinti modeliavimą. Panaudojus SYNOPSYS galimybę eksportuoti RTL aprašus į SystemC, iš susintezuotų Verilog struktūrinių modelių buvo gauti SystemC modeliai.

Eksperimentu buvo stengiamasi įvertinti atsitiktinės sekos papildymo algoritmus. Tuo tikslu kiekvienam modeliui buvo kuriamos atsitiktinės rinkinių sekos ir papildytos atsitiktinės sekos.

1 ir 3 lentelėse pateikti eksperimento rezultatai. Buvo pastebėta, kad C ir SystemC modelių veikimas nevisiškai vienodas, todėl pateikiami tik SystemC modelių rezultatai, nes jų veikimas artimesnis turimų susintezuotų schemų veikimui. Viršutinė atsitiktinai generuojamų rinkinių riba buvo nustatyta lygi 100000 ir, kaip matyti iš lentelių, keliais atvejais ji buvo pasiekta.

Iš lentelių matyti, kad du atsitiktinės sekos papildymo algoritmai (*akt* ir *neakt*) duoda geresnius rezultatus nei atsitiktinis generavimas (*rnd*). Trečiasis papildymo algoritmas (*kiekv*) beveik visais atvejais gerokai atsilieka. Atrenkama testinė seka mažai pailgėja.

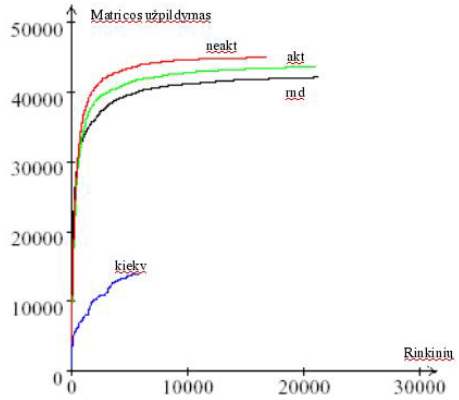
Gali kilti įtarimas, kad pabaigos sąlyga per anksti nutraukia atsitiktinio generavimo algoritmą ir palyginimas nekorektiškas. Tačiau taip nėra. 3 pav. parodyta B10SC schemos matricos užpildymo dinamika, iš ko aiškėja, kad papildymo algoritmai lenkia atsitiktinį. Grafikai pasibaigia, kai atrenkamas paskutinis rinkinys (lentelėse pateiktas visas patikrintų rinkinių skaičius). Kitų schemų rezultatai analogiški.

Norint nustatyti sukurtų testų taikymo vėlesniuose projektavimo etapuose galimybes, sukurti testai buvo įvertinti CADENCE Verifault programa. 2 lentelėje pateiktas testų pilnumas. B05, B13, B14 ir B15 schemoms gauti SystemC modelių, tiksliai atitinkančių susintezuotas Verilog schemas, nepavyko, todėl šių schemų testų pilnumas nebuvo vertinamas. Įterpiant rinkinius į atsitiktinę seką buvo galima tikėtis, kad įterpti rinkiniai

papildys matricą, bet netikrins gedimų susintezuotoje schemoje, tačiau testo pilnumas nesumažėjo nė vienu atveju, o B12SC schemai netgi padidėjo, taigi galima teigti, kad dviejų algoritmų papildomai generuojami rinkiniai nepablogina atsitiktinai kuriamos aibės.

1 lentelė. Gautų testinių sekų ilgiai

Schema	*in x out * iter*	*Testo ilgis*			
		rnd	*akt*	*neakt*	*kiekv*
B01SC	3 x 2 * 24	44	43	44	51
B02SC	2 x 1 * 42	20	19	21	19
B03SC	5 x 4 * 21	152	160	130	46
B04SC	12 x 8 * 21	2524	2864	3133	1424
B05SC	2 x 36 * 10	11	11	11	11
B06SC	3 x 6 * 30	391	439	453	345
B07SC	2 x 8 * 150	281	294	348	563
B08SC	10 x 4 * 50	1029	1092	1319	1294
B09SC	2 x 1 * 72	1148	1358	1308	92
B10SC	12 x 6 * 27	2493	2570	2624	1086
B11SC	8 x 6 * 33	1518	1540	1673	1885
B12SC	6 x 6 * 100	2237	2325	1723	73
B13SC	11 x 10 * 100	2939	2892	2962	6123
B14SC	33 x 54 * 10	4509	4570	4624	3675
B15SC	37 x 70 * 1	730	884	750	1734



3 pav. B10SC matricos užpildymas

Sukuriami testai leidžia pastebėti, jog keli tos pačios schemos modeliai veikia skirtingai. Tai patvirtina prielaidą, kad tokie testai gali būti naudojami ankstesniuose projekto etapuose verifikavimui, kai schemos modeliai konvertuojami į kitą vaizdavimo būdą (Verilog, VHDL, C/C++, SystemC ir t.t.).

Išvados

1. Naudojant atsitiktinės sekos papildymo algoritmus galima gauti geresnės kokybės testines sekas. Šitaip gauti testai, vertinant jų pilnumą konkrečiai realizacijai, nenusileidžia grynai atsitiktiniams testams.

2 lentelė. Gautų testų pilnumas

Schema	*Režimas*	*Iš viso*		*Pirminiai*	
		gedimų	*%*	*gedimų*	*%*
B01SC	rnd, akt, neakt	431	91,5	208	87,4
	kiekv	430	91,3	207	87,0
B02SC	visi	308	88,0	130	78,3
B03SC	rnd, akt, neakt	1519	68,7	648	58,4
	kiekv	1511	68,	646	58.4
B04SC	rnd, akt, neakt	5262	86,0	2255	76,2
	kiekv	5204	85,0	2219	75,0
B06SC	rnd, akt, neakt	599	83,2	284	75,5
	kiekv	598	83,1	283	75,5
B07SC	visi	2317	53,7	1035	47,6
B08SC	visi	1685	88,9	747	78,6
B09SC	rnd, akt, neakt	1778	85,0	803	74,4
	kiekv	1284	61,4	570	52,8
B10SC	rnd, akt, neakt	1627	87,1	715	78,8
	kiekv	1552	83,1	682	75,2
B12SC	rnd	2356	20,8	1024	18,0
	akt	2536	22,4	1112	19,6
	neakt	2382	21,1	1034	18,2
	kiekv	2231	19,7	962	16,9

2. Pasiūlyta pabaigos sąlyga tinkama naudojimui praktikoje. Ši sąlyga leidžia anksti nutraukti neefektyvų papildymo algoritmą.

3. Pasiūlyta testų sudarymo metodika tinka schemoms su atmintimi testuoti, kai turime tik juodosios dėžės šių schemų modelius.

4. Pasiūlyti du atsitiktinės sekos papildymo algoritmai, kurie sukuria geros kokybės testinius rinkinius.

3 lentelė. Atsitiktinės sekos papildymas

Schema	*X x Y * iter*	*Matricos langelių*	*Matricos užpildymas*				*Iteracijų*			
			rnd	*akt*	*neakt*	*kiekv*	*rnd*	*akt*	*neakt*	*kiekv*
B01SC	3 x 2 * 24	9216	520	**528**	520	263	126	148	169	133
B02SC	2 x 1 * 42	7056	59	59	59	41	100	100	100	100
B03SC	5 x 4 * 21	28224	1207	**1236**	1060	327	605	633	343	207
B04SC	12 x 8 * 21	155232	45854	**47241**	47144	18178	19837	14399	22969	11409
B05SC	2 x 36 * 10	14400	170	170	170	170	100	100	100	132
B06SC	3 x 6 * 30	43200	8864	**9351**	8964	4586	3930	4430	4194	3983
B07SC	2 x 8 * 150	720000	15245	**21788**	16470	17971	4380	4034	3911	6691
B08SC	10 x 4 * 50	360000	36635	38407	**43108**	27024	13346	15368	14675	15031
B09SC	2 x 1 * 72	20736	6745	**7561**	6982	372	3758	5029	5065	220
B10SC	12 x 6 * 27	192456	42137	43615	**45052**	13941	21518	21212	17095	5937
B11SC	8 x 6 * 33	182952	44889	46004	**46096**	20221	14801	13902	14013	5791
B12SC	6 x 6 * 100	1200000	18068	**24671**	24491	647	27727	30992	13499	264
B13SC	11 x 10 * 100	4000000	75694	**86496**	76715	76269	46234	49268	49174	100000
B14SC	33 x 54 * 10	691200	171954	174537	**175574**	87474	100000	100000	100000	55938
B15SC	37 x 70 * 1	1008000	16406	26986	17205	**34623**	18502	39158	19028	25417

Literatūra

1. **Abramovici M., Breuer M. A., Friedman A. D.** Digital Systems Testing and Testable Design. Revised printing – IEEE Press, 1990.
2. **Biere A., Kunz W.** SAT and ATPG: Boolean engines for formal harware verification // Proceedings of the 2002 International Conference on Computer Aided Design (ICCAD'02). – IEEE, 2002. – P.782-785.
3. **Sahraoui Z., Six P., Bolsen I., De Man H.** Search Space Reduction Through Clustering in Test Generation // Proceedings of the European Design Automation Conference with EURO-VHDL'95 (EURODAC'95). – IEEE, 1995. – P. 242 – 247.
4. **David R.** Random Testing of Digital Circuits: Theory and Applications. – Marcel Dekker, Inc., 1998.
5. **Behm M., Ludden J., Lichtenstein Y., Rimon M., Vinov M.** Industrial Experience with Test Generation Languages for Processor Verification // Proceedings of the 41st Design Automation Conference (DAC'04). – ACM, 2004. – P. 36 – 40.
6. **Paulikas K., Šaulys V., Šeinauskas R.** Procedures for Selection of Input Vectors on the Algorithm Level // Informacinės technologijos ir valdymas. – Kaunas: Technologija, 2000. – Nr. 2(15). – P. 59–62.
7. **Paulikas K.** Testinių rinkinių iteracinio generavimo tyrimas // Informacinės technologijos ir mokslų integracija. Konferencijos pranešimų medžiaga. – Kaunas, 2000. – P. 82 – 85.

Pateikta spaudai 2005 03 30

V. Jusas, K. Paulikas, R. Šeinauskas. Juodosios dėžės modelių testų kūrimas // Elektronika ir elektrotechnika. – Kaunas: Technologija, 2005. – Nr. 7(63). – P. 35 – 39.

Didėjant elektroninių mikroschemų sudėtingumui, jų testavimas darosi vis brangesnis ir sudėtingesnis. Kuo anksčiau aptinkamos klaidos, tuo pigiau jas ištaisyti. Pradiniuose projektavimo etapuose yra tik aukšto abstrakcijos lygio modeliai. Aprašomuose tyrimuose į abstraktų modelį žiūrima kaip į „juodąją dėžę", nesigilinant į schemos veikimą. Naudojami algoritmai remiasi modeliavimu ir nepriklauso nuo naudojamos projektavimo aplinkos. Stengiamasi sudaryti testinę seką, kurią būtų galima naudoti ankstyvuose projektavimo etapuose, o vėliau pritaikyti prie konkrečios realizacijos. Tyrimai rodo, kad gautos sekos yra gerokai trumpesnės už atsitiktinai sukuriamas, tačiau išlaiko panašią testo kokybę. Il. 3, bibl. 7 (lietuvių kalba; santraukos lietuvių, anglų, rusų k.).

V. Jusas, K. Paulikas, R. Šeinauskas. Test Design for Black-Box Models // Electronics and Electrical Engineering. – Kaunas: Technologija, 2005. Nr. 7(63). – P. 35 – 39.

Microelectronic circuits are becoming more complex and their testing is more and more complex and expensive. If errors are detected early it is cheaper to correct them. Only high abstraction level models are available at early project stages. High level model is viewed as "black-box" in our research, not looking to its internal design. Algorithms used are based on simulation techniques and are independent of design environment. It is attempted to create test sequence, which be usable in early project stages, and later to adopt it to particular requirements. Experiments show that obtained sequences are shorter than random generated, but have similar test quality. Ill. 3, bibl. 7 (In Lithuanian; summaries in Lithuanian, English, Russian).

В. Юсас, К. Пауликас, Р. Шейнаускас. Разработка тестов моделей чёрного ящика // Электроника и электротехника. – Каунас: Технология, 2005. – №. 7(63). – С. 35 – 39.

Микросхемы становятся всё сложнее, а их тестирование всё более сложной и дорогой задачей. Исправление ошибок на ранних этапах проекта намного дешевле. На ранних этапах проекта доступны лишь абстрактные модели. В наших исследованиях эти модели рассматриваются как чёрные ящики, не интересуясь их внутренней структурой. Используемые алгоритмы основаны на моделировании и не зависят от среды проектирования. Стараемся получить тестовую последовательность, которую можно использовать на ранних этапах проекта, а позже изменять в зависимости от конкретной реализации. Эксперименты показывают, что получаемые последовательности короче, но сохраняют похожие качества, как и случайные последовательности. Ил. 3, библ. 7 (на литовском языке; рефераты на литовском, английском и русском яз.).