ISSN 1392 - 1215 ELEKTRONIKA IR ELEKTROTECHNIKA. 2003. Nr. 5(47)

T 171 MIKROELEKTRONIKA

Programuojamųjų lustų testavimo metodai

V. Abraitis, E. Bareiša

Programinės įrangos katedra, Kauno technologijos universitetas
Studentų g. 50, LT-3031 Kaunas, Lietuva, tel. 300627, el.paštas abravida@elen.ktu.lt, tel. 300361,edas@soften.ktu.lt

R. Benisevičiūtė

Taikomosios elektronikos katedra, Kauno technologijos universitetas
Studentų g. 50, LT-3031 Kaunas, Lietuva, tel. 300282, el.paštas rita@soften.ktu.lt

Rinkoje pasirodžius FPGA (*Field Programmable Gate Array*) ir CPLD (*Complex Programable Logic Device*) programuojamosios logikos lustams, sutrumpintai vadinamiems PLL, atsirado galimybė projektuoti mažesnių matmenų spausdintines plokštes. Vis didėjanti programuojamojo lauko apimtis, universalumas ir sparčiai mažėjanti PLL kaina didina šio tipo schemų populiarumą ir paplitimą. Anksčiau norimai loginei funkcijai atlikti reikėdavo keleto ar keliolikos lustų, o dabar naudojamas vienas FPGA ar CPLD lustas. Programuojamosios logikos lustas gali būti pasirinktas priklausomai nuo reikiamo jo dydžio: loginių elementų ir trigerių skaičiaus; reikiamos maitinimo ir signalų įtampos (1,5; 3,.5; 5V).

Šio tipo lustų naudojimo teigiamybės:

1. sutaupoma nemažai spausdintinės plokštės ploto, nes sudėtingą skaitmeninę projekto dalį galima gauti naudojant vieną lustą. Taip suprojektuojama kompaktiška spausdintinė plokštė su viena logine mikroschema;
2. galima sumažinti projektavimo trukmę, kadangi spausdintinė plokštė gali būti projektuojama ir gaminama lygiagrečiai su skaitmeninės dalies projektavimu;
3. pagamintos plokštės atliekamą funkciją galima lengvai pakeisti perprogramavus CPLD arba FPGA atminties lustą, tam nereikia jos iš naujo projektuoti ir gaminti; padarius klaidą, užtenka perprogramuoti lustą. Kiekviename projektavimo etape galima keisti schemos funkciją;
4. galima padidinti schemos veikimo taktinį dažnį ir apdorojamų duomenų srautą; galima paspartinti ir supaprastinti skaitmeninio įtaiso projektavimą panaudojant jau sukurtus tipinių funkcinių mazgų bibliotekos elementus;
5. projektuotojo kūrinys yra saugus, niekas negali jo nukopijuoti.

Plačiai naudojamos šios XILINX gaminamos FPGA lustų šeimos: XC4000, XC4000XL, XC4000XV, Virtex, Spartan, Virtex II, Spartan IIE ,Spartan II, Virtex II Pro, taip pat CPLD lustų šeimos: XC9500, XC9500XL, XC9500XV, CoolRunner, CoolRunner II. Kiekviena lustų šeima – tai plati lustų įvairovė, apimanti loginius elementus, trigerius, daugintuvus, atminties elementus, įėjimų ir išėjimų prievadus, taip pat ir skirtingus korpusus. Virtex II Pro programuojamosios logikos lustuose integruoti net PowerPC mikroprocesoriai.

Šiuolaikinių elektroninių schemų projektavimas jau nėra susijęs su konkrečia elementų baze. Programinės priemonės tarsi suvienodina įvairias programuojamųjų struktūrų architektūras. Tik paskutiniame projektavimo etape pasirenkamas konkretus lustas ir jo elementų bazė. Programuojamosios logikos matricų gamintojų tiekiamos programinės įrangos pakanka visai schemos projektavimo eigai, pradedant schemos kodavimu ar elgsenos aprašymu ir baigiant modeliavimu, mikroschemos programavimu ir verifikavimu.

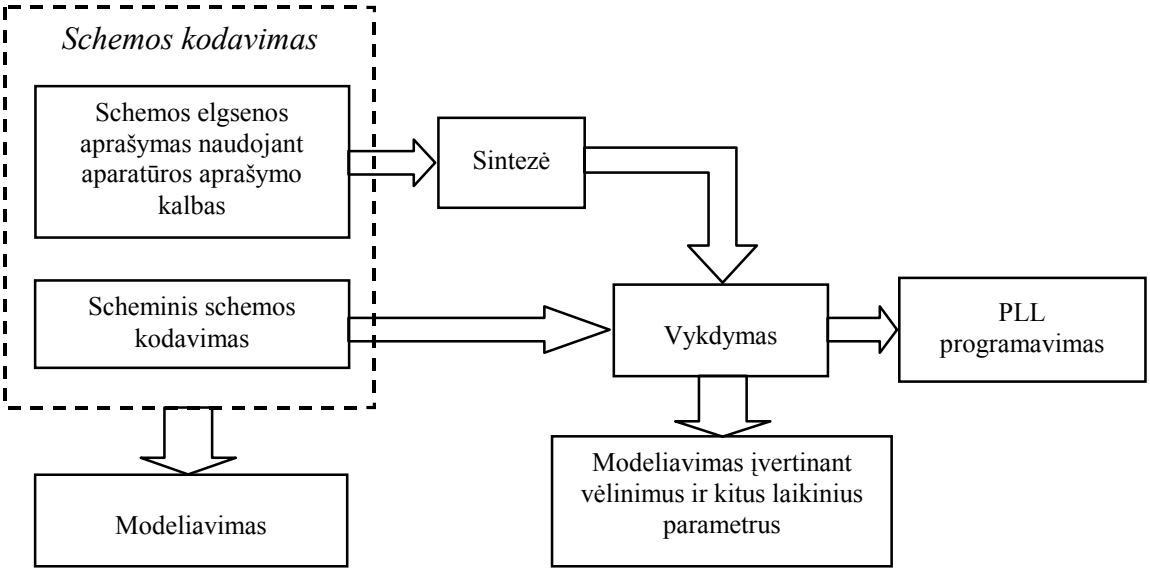
Šiuo metu vis dar naudojamos brangios, didelės, sunkiai pataisomos ir modifikuojamos spausdintinės plokštės. Tokias plokštes galima perprojektuoti panaudojant pažangią PLL technologiją. Galimi du pagrindiniai perprojektavimo būdai. Jie naudojami atsižvelgiant į turimą informacija apie senąją schemą ir projektuojamos schemos dydį.

Jei schema nėra didelė ir žinomas jos aprašas, tuomet, naudojant atitinkamą programinę įrangą, tą pačią schemą galima užkoduoti ir, pasirinkus PLL, ją užprogramuoti. Taigi tam tikrą loginę funkciją atliekantis loginių lustų rinkinys pakeičiamas vienu PLL.

Jei schema yra didesnė arba žinoma tik jos elgsena, t.y. žinomi tik įėjimo signalai ir schemos reakcija į juos, geriau naudoti aparatūros aprašymo kalbas: VHDL, Verilog arba ABEL. Toks loginės schemos aprašas sintezuojamas pasirinktoje FPGA ar CPLD lustų šeimų elementų bazėje. Taip gaunamas schemos aprašas, kurį galima naudoti programuojant PLL. Projektavimo, panaudojant PLL, procesas pavaizduotas 1 pav.

Naudojant lauku programuojamas logines matricas (FPGA), galima užprogramuoti reikiamą schemos funkciją, sumažinant kainą, sutrumpinant projektavimo trukmę ir lengvai darant pakeitimus ir pataisymus [1]. Todėl daugelis prototipų ir išbaigtų įtaisų dabar kuriami naudojant programuojamuosius lustus net ten, kur anksčiau buvo galima naudoti tik programinę įrangą.

Plačiai paplitus programuojamiesiems lustams, labai svarbūs tapo patikimumo ir testavimo klausimai. Sparčiai tobulėjant integrinių schemų technologijoms labai padidėjo loginių elementų tankis ir loginių elementų bei išorinių kontaktų skaičiaus santykis.



1 pav. Projektavimo procesas naudojant programuojamosios logikos matricas

Programuojamųjų lustų technologija šiuo metu yra įvairiapusė ir apima kelis lygius: architektūrą (logika, atmintis), aprašymo lygį (funkcinis aprašas, loginiai elementai, tranzistoriai), klaidų modelį (konstantinės klaidos, trumpieji sujungimai, neprijungtieji mazgai, atminties klaidos). Dėl šių priežasčių įprasti ASIC lustų testavimo metodai dažnai netinka programuojamiesiems lustams tirti. Todėl reikia kurti naujus testavimo metodus.

Galimi du programuojamųjų lustų testavimo scenarijai: a) gamintojas pats testuoja pagamintą lustą; b) vartotojas testuoja lustą, kai šis užprogramuojamas vykdyti konkrečią funkciją.

Gamintojo testas pagrįstas skaldymo ir valdymo principu. Kadangi programuojamieji lustai sudėtingi ir gali turėti daug įvairių komponentų, bendras testas suskaidomas į kelias testines procedūras. Kiekviena procedūra testuoja atskirą lusto komponentų grupę ir dėl šių procedūrų gaunamas visiškai patikrintas programuojamasis lustas.

Atskiri testavimo metodai skirti šių programuojamųjų lustų dalių tyrimui:

- konfigūruojami loginiai blokai [2,3];
- konfigūruojamų loginių blokų tarpusavio sujungimai (tinklas) [4];
- atminties ląstelės [5].

Tuo tarpu vartotojo naudojami testai nėra skirti visam programuojamam lustui patikrinti. Testuojama tik ta lusto dalis, kuri naudojama reikiamai funkcijai atlikti. Tokie vartotojo testai paprastai pagrįsti klasikiniais metodais, dažniausiai testinių sekų generatoriais, panaudojant modifikuotus loginio lygio schemos aprašus [6]. Naudodami šiuos testavimo metodus ir norėdami sužinoti, kurią schemos dalį patikriname, t.y. kiek galimų klaidų tikriname sukurtomis testinėmis sekomis, reikia apibrėžti galimas klaidas, nustatyti ir pašalinti perteklines klaidas. Tai ekvivalentinės, konstantinės klaidos bei galimos klaidos nenaudojamose lusto dalyse. Kadangi testinių sekų generatoriai paprastai neįvertina atminties elementų gedimų, siekiant įvertinti visas klaidas, schemą dažnai tenka modifikuoti ir pakeisti ją analogiškai funkcionuojančiu schemos modeliu. Pakeitus schemą modeliu, kartu išvengiama ekvivalentinių ir konstantinių klaidų. Sudarant testinėms sekoms generuoti tinkamus schemos modelius, pirmiausia reikia išnagrinėti programuojamųjų lustų architektūrą.

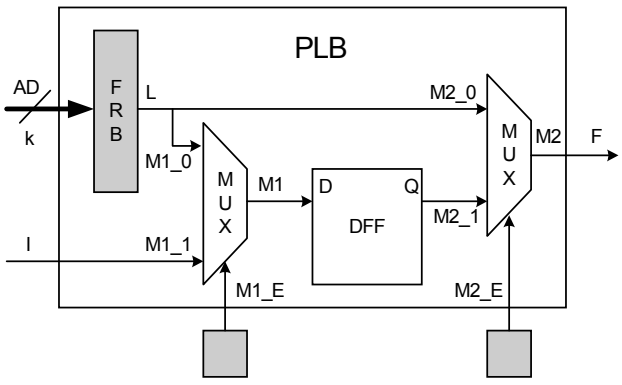
Programuojamųjų lustų architektūra

Paprastai kiekviena programuojamųjų lustų šeima pasižymi specifinėmis savybėmis. Dažniausiai programuojamieji lustai būna sudaryti iš programuojamųjų loginių blokų, t.y. PLM, kurių dydis $N \times M$ (dažniausiai N ir M būna lygūs), ir programuojamųjų įėjimų ir išėjimų blokų. Visa tai sujungta programuojamuoju ryšių tinklu.

Nagrinėsime SRAM technologijos programuojamuosius lustus, kurie programuojami į lusto konfigūracinės atminties ląsteles (KAL) įrašant reikiamas vertes. Iš esmės visų tipų programuojamųjų lustų blokai (PLB) savo vidine struktūra yra panašūs (2 pav.). Pilkos spalvos blokai žymi konfigūracines atminties ląsteles.

Programuojamieji loginiai blokai paprastai būna sudaryti iš trijų pagrindinių dalių:

- funkciją vykdančio bloko (FRB);
- multiplekserių;
- vieno D tipo trigerio.



2 pav. Programuojamosios logikos bloko (PLB) modelis

Funkciją vykdantis blokas gali būti užprogramuotas taip, kad atliktų k įėjimų turinčią kombinacinę funkciją arba būtų panaudotas kaip 2^k bitų RAM atmintis. Kuriant kombinacinę logiką, į konfigūracinę atmintį įrašoma 2^k eilučių turinti teisingumo lentelė; čia k yra FRB įėjimų skaičius. Priklausomai nuo FRB įėjimuose esančios vertės išrenkamas konfigūracinės atminties adresas ir išėjime suformuojama tuo adresu saugoma atminties vertė, t.y. gaunamas funkcijos rezultatas. Taigi, FRB gali vykdyti bet kurią kombinacinę funkciją, turinčią ne daugiau kaip k argumentų. Programuojant lustus, į FRB atstojančią konfigūracinę atmintį įrašomos atliekamos funkcijos bitų sekos, atitinkančios teisingumo lentelę. PLB vidiniai sujungimai tarp FRB ir D trigerio taip pat kontroliuojami atitinkamomis KAL. PLB supanti ir jungianti sujungimų struktūra sudaryta iš tranzistorių, kurie taip pat kontroliuojami per KAL.

Apskritai lustai programuojami į jų konfigūracinę atmintį įkraunant reikiamą programą, aprašytą kaip bitų seka. Norint išsaugoti kiekvieną tokios programos bitą, į atitinkamą konfigūracinės atminties ląstelę konfigūruojami PLB, lusto įėjimų ir išėjimų blokai ir ryšiai tarp jų. Taip užprogramuotas lustas vykdo jam skirtą funkciją, kuri paprastai vadinama konfigūracija.

Šiuo metu faktiškai visų programuojamųjų lustų, gaminamų naudojant SRAM atmintį, kaip Virtex™-II [7], ORCA™-4 [8], APEX™-II [9], PLB struktūra iš esmės yra panaši į tą, kuri parodyta 2 pav. Tik šiuo atveju parinkta sudėtingesnė struktūra. Pavyzdžiui, XILINX Virtex™-II šeimos programuojamų lustų PLB blokai turi keturis vienodus sluoksnius. Kiekvienas sluoksnis turi po du keturių įėjimų FRB, valdymo logiką, atskirą logikos posistemį aritmetinėms funkcijoms atlikti, komutuojančiuosius multiplekserius bei du trigerius. Kiekvienas keturių įėjimų FRB gali būti naudojamas reikiamai funkcijai atlikti; tai gali būti 16-os bitų RAM atmintis arba 16-os bitų postūmio registras.

Lustų testavimas

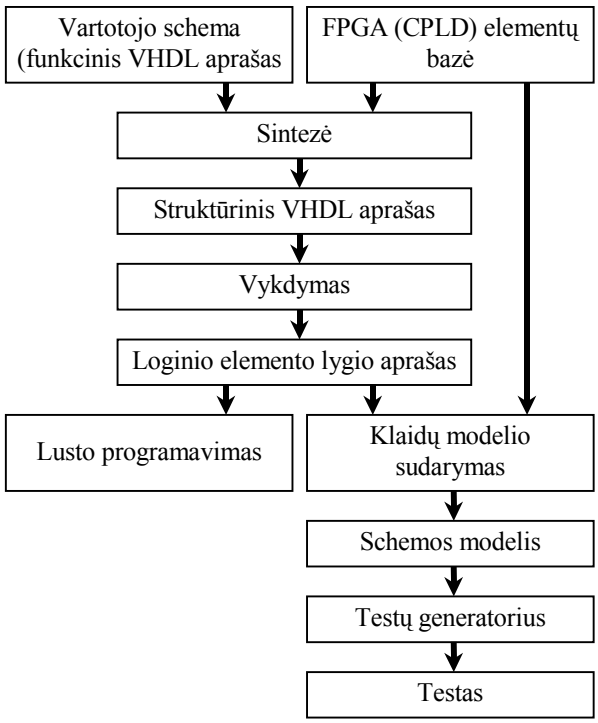
Paprastai testuojama lustą daug kartų perprogramuojant skirtingomis konfigūracijomis ir kiekvienai konfigūracijai pritaikant atitinkamą testinę seką.

Taigi testinę grupę (TG) sudaro kelios konfigūracijos ir joms skirtos testinės sekos:

$$TG = \{(C_1 \cdot S_1), (C_2 \cdot S_2), ..., (C_n \cdot S_n)\};$$

čia C_i yra i-toji konfigūracija, o S_i - jai skirta testinė seka. Paprastai testinė seka būna daug trumpesnė nei konfigūracinė seka [10].

Lyginant gamintojo ir vartotojo testines procedūras, reikia pabrėžti, kad jų tikslai visiškai skirtingi. Po pagaminimo programuojamasis lustas turi būti visiškai patikrintas nepriklausomai nuo to, kokia konfigūracija vėliau bus naudojama. Siekiant paspartinti patikrą, testinių sekų generatoriai turi sugeneruoti kuo mažiau ir kuo trumpesnes testines sekas, patikrinančias visą programuojamąjį lustą. Vartotojo testinės procedūros tikslas visiškai kitoks: šiuo atveju nereikia patikrinti viso programuojamojo lusto, užtenka patikrinti tik tą dalį, kurią naudoja konkreti konfigūracija. Todėl šis uždavinys supanašėja į tradicinį ASIC schemų testų generavimą.



3 pav. Testų generavimo proceso eiga

Čia taip pat galima naudoti tas pačias technines ir programines priemones, tačiau elektroninių schemų klaidų modeliai netinka programuojamiesiems lustams. Be to, tradiciniams schemų modeliams sukurtos testinės sekos patikrins tik nedidelę programuojamąjame luste užprogramuotos schemos dalį [11,12]. Šiuo atveju schemos aprašai prieš programuojant lustą ir po jo užprogramavimo yra skirtingi. Tokio tipo testai neįvertina vidinės programuojamųjų lustų fizinės struktūros.

Klaidų modelis

Panagrinėsime PLB patikrinimo galimybes, įvertindami jų vidinius sujungimus ir kombinacinę logiką. Remiantis 2 pav. pateikta PLB struktūra, schemos gedimai gali pasitaikyti FRB adresų (AD), atminties ląstelių išėjimų (FRB), FRB išėjimo (L), multiplekserių įėjimų (M1_0, M1_1, M2_0, M2_1) ir išėjimų (M1, M2) signaluose. Nustačius šias klaidas, galima išryškinti skirtumus tarp klasikinio ASIC schemų ir programuojamųjų lustų klaidų modelių. Kaip pavyzdį galima pasirinkti schemą, vykdančią A∩B∪C funkciją. Šiai schemai sudaryti užteks vieno PLB, o į FRB bus įkrauta pirmoji teisingumo lentelė. Remiantis pateiktu PLB modeliu ir schema, aiškiai matyti, kad aparatūriniame lygmenyje šios schemos nesutampa, taip pat nesutampa ir galimos šių schemų klaidos. Todėl testas, sudarytas remiantis tradiciniais metodais ir modeliais, patikrins tik dalį programuojamajame luste užkoduotos schemos.

Siekiant parodyti programuojamųjų lustų gedimų patikrinimo problemos aktualumą, buvo atliekamas eksperimentas, kai testas sudaromas vienai schemai, aprašytai naudojantis dviem skirtingais struktūriniais aprašais.

Eksperimentui buvo naudojamos SCAS-85 testinės schemos [13]. Kiekviena schema buvo sintezuota naudojant jų originalią elementinę bazę ir „Synopsys“ sistemos elementų bazę. Šiems dviem skirtingiems schemų aprašams buvo sudaryti testai. Vėliau schemai su originalia elementine baze skirtas testas naudotas schemai su „Synopsys“ sistemos elementų baze ir atvirkščiai. Atlikus tyrimus pastebėta, kad tos pačios elgsenos, bet skirtingo aprašo schemoms netinka vienas ir tas pats testas, nes jis dažnai nebegali patikrinti visos schemos. Testuojamų schemų parametrai parteikti 1 lentelėje, gauti rezultatai – 2 lentelėje.

1 lentelė. Testuojamų schemų parametrai

Schemos	Įėjimų	Išėjimų	Gedimų skaičius	
			Originali elementų bazė	„Synopsys“ elementų bazė
C432	36	7	1052	762
C880	60	26	1526	1608
C2670	233	140	5646	2732
C5315	178	123	13816	6336
C7552	207	108	12247	4368

2 lentelė. Testavimo rezultatai - patikrintų gedimų skaičius

Patikrintų gedimų skaičius	Originalios elementų bazės testas		“Synopsys” elementų bazės testas	
Testuojama schema	Originali elementų bazė	„Synopsys“ elementų bazė	Originali elementų bazė	“Synopsys” elementų bazė
C432	1052 gedimai 100%	755 gedimai 99,05%	1025 gedimai 97,44%	762 gedimai 100%
C880	1526 gedimai 100%	1587 gedimai 98,69%	1510 gedimų 98,95%	1608 gedimai 100%
C2670	5646 gedimai 100%	2664 gedimai 97,51%	5586 gedimai 98,94%	2732 gedimai 100%
C5315	13816 gedimų 100%	6323 gedimai 99,79%	13641 gedimas 98,74%	6336 gedimai 100%
C7552	12247 gedimai 100%	4365 gedimai 99,93%	12008 gedimai 98,05%	4368 gedimai 100%

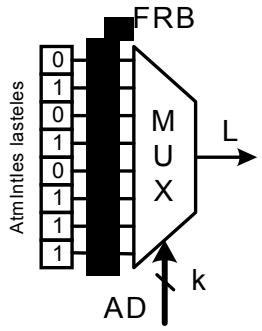
Iš pateiktų testavimo rezultatų matyti, kad vienai elementų bazei sudarytas testas netikrina apie 2,5% gedimų kitoje elementų bazėje. Kadangi elementų bazės panašios, nepatikrintų gedimų yra daug. Literatūroje [10] nurodoma, kad, naudojant programuojamųjų lustų elementų bazę, nepatikrinamų gedimų skaičius gali išaugti iki 20%

Norint panaudoti tas pačias programines testų generavimo priemones ir sudaryti išsamesnį lusto testą tam tikrai konfigūracijai, testą reikia sudaryti modifikuotam PLB modeliui. Toks modelis turi atspindėti visus galimus PLB naudojamos aparatūros gedimus ir būti lengvai testuojamas.

Siekiant įvertinti FRB išėjimo (L) adresų (AD) ir atminties ląstelių išėjimų (FRB) signalų galimus gedimus visas FRB keičiamas multiplekseriu. Signalu AD išrenkama atitinkamos FRB atminties ląstelės reikšmė. Taip gaunamas FRB modelis, kuris veikia taip pat kaip FRB, tačiau leidžia įvertinti jo gedimus ir lengvai sukurti testinę seką. Toks modelis pateiktas 3 pav.

Testo kūrimas naudojant siūlomą klaidų modelį

Norint sukurti siūlomą klaidų modelį ir sudaryti testines sekas, nuodugniau patikrinančias programuojamųjų lustų schemą, gali būti panaudotas 4 pav. pateiktas testo kūrimo procesas. Čia sintezei ir realizacijai naudojamos „Xilinx“ programinės priemonės [14].



4 pav. A∩B∪C funkciją vykdančio FRB modelis

Klaidų modelis sudaromas naudojant loginių elementų lygio schemos aprašą ir elementų bazę, būtent čia FRB pakeičiami multiplekseriais. Testinei sekai kurti naudojamos integruotosios „Synopsys“ projektavimo sistemos ATPG galimybės [15].

Išvados

1.Pateiktas klaidų modelis naudotinas programuojamiesiems lustams testuoti. Modelis leidžia taikyti klasikinius ASIC schemoms skirtus testų generavimo būdus ir testavimo priemones.

2. Modelis sukurtas, siekiant gauti išsamesnius testus sukurtoms programuojamųjų lustų schemoms. Naujų testų sudarymo metodų ir modelių poreikis išnagrinėtas ir įrodytas eksperimentiškai.

Literatūra

1. **Brown S.D., Rose J**. FPGA and CPLD Architectures: A Tutorial // IEEE Design and Test of Computers. – 1996. - Nr. 13-2. - P.42-57.
2. **Abramovici M., Stroud C.** BIST-Based Test and Diagnosis of FPGA Logic Blocks // IEEE Transactions on VLSI Systems. – 2001. – No.9-1. - P.159-172.
3. **Zhao L., Walker D.M., Lombardi F.** Detection of Bridging Faults in Logic Resources of Configurable FPGAs Using Iddq // International Test Conference. – 1998. – P.1037-1046.
4. **Renovell M., Portal J.P., Figueras J., Zorian Y.** Testing the Interconnect of RAM-Based FPGAs // IEEE Design & Test of Computers. – 1998. - P.45-50.
5. **Renovell M., Portal J.M., Figueras J., Zorian Y.** SRAM-Based FPGA's: Testing the LUT/RAM Modules // IEEE International Test Conference. – 1998. - P.1102-1111.
6. **Renovell M., Portal J.M., Faure P., Figueras J., Zorian Y.** Analyzing the Test Generation Problem for an Application-Oriented Test of FPGAs // IEEE European Test Workshop. – 2000. - P.75-80.
7. Xilinx Inc. Virtex™-II Platform FPGA Handbook. - 2001, Prieiga per internetą: http://www.xilinx.com/.
8. Lattice Semiconductor Co. - ORCA Series 4 FPGAs Data Sheet. – 2002. Prieiga per internetą: http://www.latticesemi. com/.
9. Altera Co. APEX™-II Programmable Logic Device Family.- 2001, gruodis [žiūrėta 2003-03-11]. Prieiga per internetą: http://www.latticesemi.com/
10. **Rebaudengo M., Reorda S. M., Violante M.** A new functional Fault Model for FPGA Application-Oriented Testing. - 2002 Prieiga per internetą:http://www.cad.polito.it/pap/db/ dft2002a.pdf .
11. **Šeinauskas R.** ASIC Design Flow and Test Generation Capabilities // Informacinės technologijos ir valdymas.- ISSN 12392 – 1215. 2003, Nr.1. – P.55.
12. **Renovell M., Figueras J., Zorian Y.** Test of RAM-Based FPGA: Methodology and Application to the Interconnect // IEEE VLSI Test Symposium. – 1997. P.230-237.
13. Collaborative Benchmarking Laboratory. ISCAS'85 Benchmark Information. 1997. - Prieiga per internetą: http://www. cbl.ncsu.edu/CBL_Docs/iscas85.html.
14. Synopsys Inc. FPGA Compiler II™. 2001. - Prieiga per internetą: http://www.synopsys.com/.
15. Synopsys Inc. TestGen™ v. 4.0. – 2000. - Preiga per internetą: http://www.synopsys.com/ .

Pateikta spaudai 2003 05 02

V. Abraitis, E. Bareiša, R. Benisevičiūtė. Programuojamų lustų testavimo metodai // Elektronika ir elektrotechnika. - Kaunas: Technologija, 2003. - Nr. 5(47). - P. 43-47.

Nagrinėjamos programuojamųjų loginių struktūrų (FPGA) rūšys, jų galimybės, teigiamybės ir trūkumai. Analizuojamas programuojamų lustų gedimų modelis, kai lustas užprogramuotas vykdyti vartotojo norimą funkciją. Siūlomas gedimų modelis leidžia naudoti tradicinius testinių sekų generatorius. Eksperimentiškai ištirta, kad skirtingiems loginių schemų aprašams reikalingi skirtingi testų rinkiniai. Il. 4, bibl. 15 (lietuvių kalba;, santraukos lietuvių, anglų ir rusų k.).

V. Abraitis, E. Bareiša, R. Benisevičiūtė. The Testing Methods of Programmable Integrated Circuits // Elektronika ir elektrotechnika. - Kaunas: Technologija, 2003. - No. 5(47). - P. 43-47.

In this paper the types of structures of programmable integrated circuits (FPGA), their possibilities, advantages and faults are considered. The fault models of programmable integrated circuits are analyzed, when programmable integrated circuits are configured to implement a given application. Proposed fault model can be used with traditionally automatic test sequence generators. The results of experiment show that for different synthesis of the same circuit we need different test sequences. Ill. 4, bibl. 15 (in Lithuanian; summaries in Lithuanian, English and Russian).

В. Абрайтис, Э. Барейша, Р. Бенисявичюте. Методы тестирования программируемых лустов // Электроника и электротехника. - Каунас: Технология, 2003. – No. 5(47). - С.43-47.

Рассматриваются разновидности, возможности и особенности программируемых логических структур. Проведен анализ модели программируемых интегральных схем (ИС), когда ИС запрограммирована выполнять соответственную функцию. Предлагаемая модель неисправностей может быть использована с традициоными генераторами тестовых последовательностей. Экспериментальные исследования показали, что применение различных моделей логических схем требует соответствующих тестовых наборов. Ил. 4, библ. 15 (на литовском языке; рефераты на литовском, английском и русском яз.).

DOI: 10.5755/j02.eie.11240