*ISSN 1392 - 1215 ELEKTRONIKA IR ELEKTROTECHNIKA. 2005. Nr. 6(62)*

*T 171 MIKROELEKTRONIKA*

# Programuojamosios logikos elemento gedimo modelis

**V. Abraitis, E. Bareiša**

*Programų inžinerijos katedra, Kauno technologijos universitetas,*
*Studentų g. 50, LT-51368 Kaunas, Lietuva, tel.+370 37 300361; el.p.: abravida@elen.ktu.lt; edas@soften.ktu.lt*

## Įvadas

Šiuo metu pramonėje labai specializuotų įrenginių poreikis vis didėja, bet tokių įrenginių paprastai būna nedaug. O ką nors gaminti mažais kiekiais niekad nebuvo ekonomiškai naudinga. Dėl šios priežasties rinkoje vis dažniau pasirodo įvairių plačios paskirties gaminių, neturinčių konkrečios atliekamos funkcijos. O įrenginio funkciją leidžiama pasirinkti pačiam vartotojui. Tokių įrenginių galima gaminti didesnį kiekį, o tai, vertinant ekonominiu aspektu, yra kur kas naudingiau. Vartotojas paprastai įsigyja tik aparatinę dalį ir dar galbūt kažkiek programinės įrangos tai aparatinei daliai palaikyti, o visa kita susikuria pats arba leidžia tai atlikti rangovui ir šitaip gauna galutinį specializuotą produktą, visiškai atitinkantį jo poreikius. Vieni tokių produktų yra FPGA (Field Programmable Gate Array) ir CPLD (Complex Programable Logic Device) programuojamosios logikos lustai, sutrumpintai vadinami PLL. Vis didėjanti programuojamojo lauko apimtis, greitaveika, universalumas ir sparčiai mažėjanti PLL kaina padidino šio tipo schemų populiarumą ir paplitimą. Programuojamosios logikos lustas gali būti pasirinktas priklausomai nuo jo dydžio, loginių elementų ir trigerių skaičiaus, reikiamos maitinimo ir signalų įtampos.

Šiuo metu PLL naudojami civilinėje pramonėje įvairiems gamybos procesams valdyti, karinėje pramonėje bei medicinoje. Dėl šių priežasčių programuojamiesiems lustams labai svarbūs tapo patikimumo ir testavimo klausimai. O dėl vidinės PLL struktūros specifikos įprasti ASIC lustų testavimo metodai faktiškai netinka programuojamiesiems lustams. Todėl reikia kurti naujus arba kokiu nors būdu pritaikyti esamus ASIC schemoms skirtus testavimo metodus.

## Programuojamųjų lustų testavimo scenarijai

Testuojant programuojamuosius lustus galimi du testavimo scenarijai:

1. gamintojas pats testuoja pagamintą lustą;
2. vartotojas testuoja lustą po to, kai šis užprogramuojamas vykdyti konkrečią funkciją.

Gamintojo testas pagrįstas skaldymo ir valdymo principu. Kadangi programuojamųjų lustų struktūra labai sudėtinga ir galima plati komponentų įvairovė, bendras testas suskaidomas į kelias testines procedūras. Kiekviena procedūra skirta atskirai komponentų grupei testuoti. O šių procedūrų rezultatas – išsamiai patikrintas programuojamasis lustas.

Atskiri testavimo metodai yra skirti šioms programuojamųjų lustų dalims tikrinti:

1. programuojamiesiems loginiams blokams (PLB) [1,2];
2. programuojamųjų loginių blokų tarpusavio sujungimams (sujungimų tinklui) [3];
3. atminties ląstelėms [4].

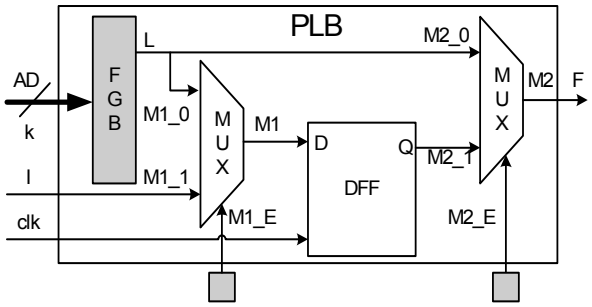
Gamintojai paprastai parduoda jau patikrintus ir funkcionuojančius įrenginius, tačiau sandėliavimo, transportavimo ar eksploatavimo metu programuojamosios logikos įrenginiuose gali atsirasti gedimų. Dėl šios priežasties vartotojo testai taip pat yra aktualūs ir reikalingi. Tačiau vartotojo naudojami testai nėra skirti visam PLL patikrinti. Testuojama tik ta lusto dalis, kuri naudojama reikiamai funkcijai atlikti. Todėl šis uždavinys darosi panašus į tradicinį ASIC schemų testų generavimą. Šiam uždaviniui spręsti būtų galima panaudoti tas pačias technines ir programines priemones, tačiau tradiciniai elektroninių schemų gedimų modeliai netinka programuojamiesiems lustams. Be to, tradiciniams schemų modeliams sugeneruotos testinės sekos patikrina tik nedidelę programuojamajame luste užprogramuotos schemos dalį, nes schemų aprašai yra skirtingi. Tai buvo patikrinta eksperimentiškai, o rezultatai paskelbti ankstesniame straipsnyje [5]. Tokio tipo testai neįvertina vidinės programuojamųjų lustų fizinės struktūros [6,7]. Kadangi testinių sekų generatoriai paprastai neįvertina atminties elementų gedimų, siekiant įvertinti visus gedimus, schemą tenka modifikuoti ir pakeisti ją taip pat funkcionuojančiu schemos modeliu. Sudarant testinėms sekoms generuoti tinkamus schemos modelius, pirmiausia reikia išnagrinėti programuojamųjų lustų architektūrą.

## Programuojamųjų lustų architektūra

Paprastai kiekviena programuojamųjų lustų šeima pasižymi savitomis specifinėmis savybėmis. Dažniausiai programuojamieji lustai būna sudaryti iš programuojamųjų loginių elementų, kitaip vadinamų PLB (programuojamasis loginis blokas). Programuojamąjį loginį lustą sudaro PLB matrica, kurios dydis $N \times M$, ir programuojamieji įėjimų ir išėjimų blokai. Visa tai tarpusavyje sujungta programuojamuoju ryšių tinklu.

Konkrečiu atveju nagrinėsime SRAM technologijos programuojamuosius lustus, kurie programuojami į lusto

konfigūracinės atminties ląsteles (KAL) įrašant reikiamas vertes. Iš esmės visų tipų programuojamųjų lustų PLB savo vidine struktūra yra panašūs (1 pav.).




**1 pav.** Supaprastintas programuojamosios logikos bloko modelis

Programuojamieji loginiai blokai paprastai būna sudaryti iš trijų pagrindinių dalių: funkcijos generatoriaus bloko (FGB), multiplekserių ir D tipo trigerio.

Pilkos spalvos blokai (1 pav.) žymi konfigūracines atminties ląsteles. FGB gali būti užprogramuotas taip, kad atliktų *k* įėjimų turinčią kombinacinę funkciją. Atliekant norimą funkciją, į konfigūracinę atmintį įrašoma 2k eilučių turinti reikiamos funkcijos teisingumo lentelė; čia *k* yra FGB įėjimų skaičius. PLB vidiniai sujungimai tarp FGB ir D trigerio taip pat kontroliuojami atitinkamomis KAL ląstelėmis. PLB supanti ir jungianti sujungimų struktūra sudaryta iš tranzistorių, kurie taip pat kontroliuojami per KAL ląsteles. Apskritai lustai programuojami į jų konfigūracinę atmintį įkraunant reikiamą programą, aprašytą kaip bitų seka. Taip užprogramuotas lustas vykdo jam skirtą funkciją, kuri paprastai vadinama konfigūracija.

Šiuo metu faktiškai visų programuojamųjų lustų, gaminamų naudojant SRAM technologiją, tokių kaip Virtex–4 [8], ProASIC™ [9], ORCATM–4 [10], Stratix II [11], PLB blokų struktūra iš esmės yra panaši kaip ir parodyta 1 pav., tik sudėtingesnė. Pavyzdžiui, vieną Virtex–4 XC4VLX200–FF1513 lusto PLB sudaro du trigeriai, du FGB, keletas multiplekserių, XOR ir IR elementų. Yra specialūs signalai informacijai tiesiogiai perduoti iš vieno PLB į kitą, aplenkiant bendrą ryšių tinklą.

**Programuojamosios logikos elemento gedimų modelis**

Lustų gedimai paprastai modeliuojami, o sukurtiems modeliams yra sudaromi testai, kuriais galima tikrinti ir pačius lustus. Taip yra daroma dėl kelių pagrindinių priežasčių:

- su modeliais dažnai paprasčiau dirbti;
- modelius galima perkelti iš vienos sistemos į kitą;
- modelius galima naudoti simuliacijai ir taip išvengti išlaidų, neišvengiamų gaminant fizinį modelio atitikmenį.

Visos šios priežastys tinka ne tik gedimams, bet ir funkcionavimui modeliuoti. Gedimų modeliai parodo, ką būtent turime tikrinti, taip pat yra galimybė analizuoti šiuos modelius ir galimus gedimus. Be to, modelio efektyvumą galima išmatuoti eksperimentiškai.

Realiai lustuose pasitaiko apdorojimo defektų (nėra kontakto, parazitiniai tranzistoriai, oksido pramušimas), medžiagų defektų (stambūs defektai: skilimai, kristalo neištisumas ir negrynas paviršius – jonų migracija), nuo laiko priklausančių gedimų (suardytas dielektrikas, srovės nutekėjimas), taip pat pakavimo trūkumų (kontaktų sugadinimas, izoliacijos pažeidimas). Dažniausiai pasitaiko vieno laido fiksacijos, atviro ir trumpai sujungto tranzistoriaus, atminties, vėlinimo gedimai ir klaidos.

Vieno laido fiksacijos gedimą nusako šie požymiai: defektuotas tik vienas laidas, jo vertė konstantinė – nustatyta į loginį 0 ar į loginį 1; tokia klaida gali būti ir elemento įėjime, ir išėjime.

Vieno laido fiksacijos gedimų pagrindinės klasės, kurias gali aptikti automatiniai testinių sekų generatoriai ir simuliavimo programos, yra šios:

- potencialiai aptinkami defektai – testas parodo nežinomą būseną pirminiame išėjime;
- nustatymo defektai – defektai, neleidžiantys nustatyti pradinės defektuotos grandinės vertės;
- hiperaktyvumo defektai – defektai, sužadinantys daugelio vidinių signalų aktyvumą, nepasiekdami pirminio išėjimo;
- pertekliniai gedimai – gedimai, pasikartojantys toje pačioje grandinėje;
- nepatikrinimai gedimai – gedimai, kuriems testų generatorius negali surasti testinės sekos.

Remdamiesi jau susiformavusia ir jau klasikine tapusia ISIC schemų testavimo teorija, panagrinėkime programuojamojo lusto PLB patikrinimo galimybes, atsižvelgdami į jų vidinius sujungimus ir kombinacinę logiką. Remiantis 1 pav. pateikta PLB struktūra, schemos gedimai gali pasitaikyti FGB adresų (AD), FGB atminties ląstelių išėjimų (FGB), FGB išėjimo (L), PLB duomenų įėjimo (I), trigerio įėjimo (D), trigerio išėjimo (Q), multiplekserių įėjimų (M1_0, M1_1, M2_0, M2_1), multiplekserių išėjimų (M1, M2), multiplekserių valdymo (M1_E, M2_E), PLB išėjimo (F) signaluose. Eliminavus ekvivalentinius ir konstantinius gedimus reikėtų įvertinti tik šiuos gedimus:

- FGB bloko adresų signalų (AD);
- FGB bloko atminties ląstelių (FGB);
- FGB bloko išėjimo (L);
- Multiplekserių įėjimų (M1_0, M1_1, M2_0, M2_1);
- Multiplekserių išėjimų (M1, M2).

Nustačius šiuos gedimus galima, išryškinti skirtumus tarp klasikinių ASIC schemų ir programuojamųjų lustų gedimų modelių. Kaip pavyzdį galima pasirinkti schemą, vykdančią $A \cap B \cup C$ funkciją. Tokia schema turės tris įėjimus ir vieną išėjimą. Ją sudarys vienas IR elementas ir vienas ARBA elementas. Naudojant PLL, tokiai pačiai funkcijai vykdyti užtenka vieno PLB, į FGB bus įkrauta funkcijos teisingumo lentelė. Remiantis 1 pav. pateiktu PLB modeliu, aiškiai matyti, kad aparatiniame lygmenyje šios schemos nesutampa. Taip pat nesutampa ir šių schemų galimos klaidos. Todėl testas, sudarytas remiantis tradiciniais metodais ir modeliais, patikrins tik dalį programuojamajame luste užkoduotos schemos [5].

Siekiant parodyti programuojamųjų lustų gedimų patikrinimo problemos aktualumą, buvo atliekamas eksperimentas, kai testas sudaromas vienai schemai, aprašytai naudojantis dviem skirtingais struktūriniais aprašais. Eksperimento rezultatai pateikti 1 lentelėje [5]. Eksperimentui buvo naudojamos ISCAS–85 testinės schemos [12]. Iš pa-

teiktų eksperimento rezultatų matyti, kad vienai elementų bazei sudarytas testas netikrina apie 2,5% tos pačios schemos gedimų kitoje elementų bazėje. Atsižvelgiant į tai, kad elementų bazės panašios, nepatikrintų gedimų yra daug. Literatūroje [13] nurodoma, kad, naudojant programuojamųjų lustų elementų bazę, nepatikrintų gedimų skaičius gali išaugti iki 20 %.

**1 lentelė.** Testavimo rezultatai – nepatikrintų gedimų skaičius

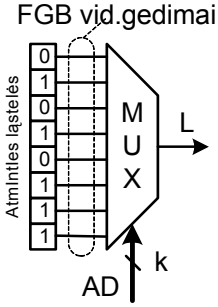
| Testuojama schema | Nepatikrintų gedimų skaičius | |
|---|---|---|
| | Originalios elementų bazės testas, pritaikytas schemai su Synopsys elementais | Synopsys elementų bazės testas, pritaikytas schemai su originaliais elementais |
| C432 | 4 (0,95%) | 13 (2,56%) |
| C499 | 34 (3,48%) | 8 (1,07%) |
| C880 | 1 (0,12%) | 21 (2,23%) |
| C1908 | 0 (0,00%) | 164 (8,81%) |
| C2670 | 39 (2,49%) | 21 (1,06%) |
| C3540 | 4 (0,17%) | 61 (1,95%) |
| C5315 | 8 (0,21%) | 66 (1,26%) |
| C6288 | 59 (0,88%) | 0 (0,00%) |
| C7552 | 12 (0,26%) | 223 (3,17%) |

**Pirmas konstantinių gedimų FGB modelis**

Aparatiškai FGB realizuotas panaudojant atmintį. Automatiniai testų generatoriai gali sudaryti testines sekas ir atminčiai patikrinti, tačiau schemos veikimo metu, po konkrečios konfigūracijos įkrovimo programuojamuosiuose lustuose FGB turinys nesikeičia ir negali keistis iki tol, kol nebus išjungtas maitinimas ar neįkrauta nauja konfigūracija (atvejai, kai FGB naudojami būtent kaip RAM atmintis, nenagrinėjami). Be to, automatiniams testų generatoriams testo sudarymo metu nėra galimybės nurodyti FGB turinio.

Kaip žinoma, atminčiai patikrinti reikalingas visiškas perrinkimas, t. y. norint patikrinti atmintį, testinę seką sudarys visos įmanomos įėjimų kombinacijos. Todėl norint panaudoti tas pačias programines testų generavimo priemones testo sudarymo metu FGB reikėtų pakeisti kažkuo, kam patikrinti taip pat reikėtų visiško perrinkimo. Tokiu tikslu sudaromas PLB modelis, kuris turi atspindėti visus galimus PLB naudojamos aparatūros gedimus ir būti lengvai testuojamas. Siekiant įvertinti FGB išėjimo (L) adresų (AD) ir atminties ląstelių išėjimų (FGB vid. gedimai) signalų galimus gedimus visas FGB keičiamas multipleksoriumi. Signalais AD išrenkamos atitinkamos FGB atminties ląstelių vertės. Šitaip gaunamas FGB modelis, savo funkcionavimu nesiskiriantis nuo FGB, tačiau leidžiantis įvertinti jo gedimus ir lengvai sudaryti testinę seką. Toks modelis pateiktas 2 pav.

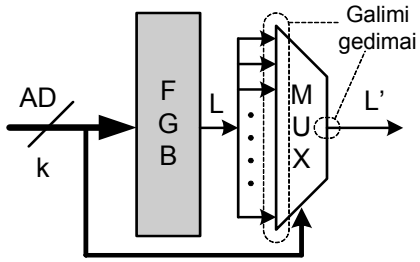
Pačios atminties ląstelės gali būti pakeistos nuosekliai įkraunamu postūmio registru. O sudarant testą šio registro galimų gedimų nederėtų tikrinti. Tuomet automatinio testų generatoriaus uždavinys taptų šiek tiek paprastesnis. Tačiau testo sudarymo metu schemos atliekama funkcija vis dar būtų neaiški, nes testas būtų sudaromas schemai be konkrečių postūmio registro verčių. Supaprastinant modelį, vietoj atminties ląstelių galima panaudoti konstantinius signalų loginius lygius. Tuomet supaprastėtų ir gedimų modelis, liktų tik FRG įėjimų (AD) ir išėjimo (L) fiksacijos į 0 ir 1 gedimai. Tačiau šis modelis turi ir trūkumą – reikia palyginti smarkiai modifikuoti schemą, kuriai sudaromas testas.




**2 pav.** A∩B∪C funkciją vykdantis FGB modelis

**Antras konstantinių gedimų FGB modelis**

Siekiant sudaryti išsamesnį lusto testą tam tikrai konfigūracijai, gali būti panaudotas kitas gedimų modelis, visiškai atitinkantis PLB funkcionavimą ir leidžiantis įvertinti galimus jo gedimus, pateiktas 3 paveiksle [14]. Čia nutraukiama FGB išėjimo grandinė ir įterpiamas multiplekseris, kurio įėjimų skaičius priklauso nuo FGB dydžio. Multiplekserio įėjimuose galimi gedimai atitinka galimus FGB gedimus. Generuojant testą, pats FGB laikomas juodąja dėže, kurios įėjimų ir išėjimų tikrinti nereikia. Juodosios dėžės viduje paliekama tradicinė ASIC schema. Tokiu būdu jau testo sudarymo metu turime PLB bloko modelį, vykdantį nustatytą funkciją, kas buvo negalima naudojant pirmąjį modelį su postūmio registru.




**3 pav.** FGB modelis

Turint tokį FGB modelį, testai generuojami tokia tvarka:

- schema sintezuojama naudojant PLL elementų bazę; gaunamas schemos aprašas, kuriame naudojami konkretūs PLB ir jau žinoma kiekvieno FGB atliekama funkcija;
- schemos aprašas išsaugomas taip, kad išliktų kiekvieno FGB vidinė struktūra;
- įterpiami multiplekseriai;
- pagal užsibrėžtą klaidų modelį nurodomi galimi gedimai;
- naudojant ATPG gaunamas testinių sekų rinkinys, kuriuo naudojantis galima patikrinti daugiau galimų programuojamojo lusto gedimų, o tai reiškia, kad gauname išsamesnį testą.

**2 lentelė** ISCAS-85 schemų konstantinių gedimų tikrinimo rezultatai

| Schema | Patikrinamų gedimų skaičius | | Class elementų bazės testo testinių rinkinių skaičius ir efektyvumas | | | Virtex elementų bazės testo testinių rinkinių skaičius ir efektyvumas | | |
|---|---|---|---|---|---|---|---|---|
| | Class | Virtex | Rink.sk. | Class | Virtex | Rink.sk. | Class | Virtex |
| C17 | 14 | 59 | 7 | 100% | 96,55% | 6 | 100% | 100% |
| C432 | 86 | 922 | 21 | 100% | 80,69% | 87 | 100% | 100% |
| C880 | 172 | 1974 | 19 | 100% | 83,59% | 93 | 100% | 100% |
| C2670 | 441 | 3551 | 49 | 100% | 92,09% | 142 | 100% | 100% |
| C5315 | 602 | 7745 | 22 | 100% | 86,95% | 135 | 100% | 100% |
| C7552 | 626 | 9062 | 41 | 100% | 89,77% | 219 | 100% | 100% |

## Eksperimentų rezultatai

Eksperimentui buvo naudojamos ISCAS–85 testinės schemos [12] ir Synopsys TetraMax programinė įranga [15]. 2 lentelėje pateikiami eksperimento rezultatai, gauti sudarant testus, skirtus konstantiniams gedimams tikrinti. Eksperimento tikslas – įrodyti, kad ventilinio lygio schemai sudarytas testas nepatikrina visų schemos gedimų, kai schema užkoduojama naudojant Virtex elementinę bazę, net tuo atveju, kai priimama sąlyga, kad gedimai gali būti tik FGB bloko įėjimuose. Eksperimentas buvo atliekamas tokia tvarka:

1. Sudaryti testai, skirti konstantiniams gedimams tikrinti ventilinio lygio schemose, užkoduotose naudojant Class elementų biblioteką (TKC).
2. Naudojant TKC testus, bandyta patikrinti schemas, užkoduotas naudojant Class ir Virtex elementų bibliotekas. Įsitikinta, kad schemai Class elementų bazėje skirtas testas nepatikrina visų patikrinamų gedimų, kai ta pati schema užkoduojama naudojant Virtex elementų bazę.
3. Sudaryti testai, skirti konstantiniams gedimams tikrinti schemose, užkoduotose naudojant Virtex elementų biblioteką (TKV).
4. Naudojant TKV testus, tikrintos schemos, užkoduotos naudojant Class ir Virtex elementų bibliotekas. Remiantis rezultatais, galima teigti, kad TKV testai yra išsamesni ir gali patikrinti schemų konstantinius gedimus tiek Class, tiek Virtex elementų bazėse.

Tačiau sudarant testus programuojamiesiems loginiams įrenginiams testo išsamumą reikėtų vertinti ne taip, kaip ventilinio lygio schemoms. Čia reiktų įvertinti ir vidines programuojamojo lusto struktūras, pavyzdžiui, FGB atminties ląsteles. Išsamiausias testas yra visiškas perrinkimas, kai visuose schemos elementų įėjimuose pateikiami visi įmanomi verčių variantai. Vertinant pagal šį kriterijų sudarytų testų išsamumas nėra jau toks didelis, kai schemos užkoduojamos naudojant Virtex elementų bazę. Siekiant padidinti testo išsamumą ir naudojamas FGB bloko modelis. 3 lentelėje pateikiamas testų išsamumo įvertinimas. Eksperimento rezultatai gauti sudarant testus, skirtus konstantiniams gedimams tikrinti.

Iš 3 lentelėje pateiktų duomenų matome, kad testiniai rinkiniai, gauti naudojant modelį, programuojamąjį lustą patikrina geriau. Tačiau susidaro įspūdis, kad efektyvumas krenta didėjant schemai ir artėja prie testo, sudaryto nenaudojant modelio. Todėl ateityje reikia detaliau ištirti ar efektyvumas mažėja dėl schemos dydžio, ar dėl sudėtingumo.

**3 lentelė.** ISCAS-85 schemų testų išsamumo įvertinimas

| Schema | Testo išsamumas | |
|---|---|---|
| | Virtex testas | Virtex + MUX testas |
| C17 | 50% | 70,83% |
| C432 | 71,1% | 83,44% |
| C880 | 63,66% | 73,02% |
| C2670 | 57,1% | 62,18% |
| C5315 | 60,66% | 62,82% |
| C7552 | 55,78% | 58,1% |

## Išvados

1. Modelis sudarytas, siekiant gauti išsamesnius testus schemoms, sudarytoms naudojant programuojamuosius lustus. Naujų testų sudarymo metodų ir modelių poreikis išnagrinėtas ir įrodytas eksperimentiškai.

2. Pateikti gedimų modeliai naudotini testuojant programuojamuosius lustu. Modelis leidžia taikyti klasikinius ASIC schemoms skirtus testų generavimo būdus ir testavimo priemones.

3. Schemos ryšiai tarp programuojamosios logikos blokų apriboja galimybes patikrinti blokus su visais galimais įėjimo poveikiais.

## Literatūra


1. **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.
2. **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.
3. **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.
4. **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.
5. **Abraitis V., Bareiša E., Benisevičiūtė R.** Programuojamųjų lustų testavimo metodai // Elektronika ir elektrotechnika, ISSN 1392 – 1215.– 2003. – Nr.5(47). – P.43–47.
6. **Šeinauskas R.** ASIC Design Flow and Test Generation Capabilities // Informacinės technologijos ir valdymas.- ISSN 12392 – 1215. – 2003.– Nr.1. – P.55.

7. **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.
8. **Xilinx Inc.** Virtex–4 Overview. – 2004. Prieiga per internetą: http://www.xilinx.com/products/virtex4/overview.htm .
9. **Actel Corporation.** ProASIC – The Nonvolatile Reprogrammable Gate Array. – 2004. Prieiga per internetą: http://www.actel.com/products/proasic/index.html .
10. **Lattice Semiconductor Co.** ORCA Series 4 FPGA Family. – 2002. Prieiga per internetą: http://www.latticesemi.com/products/fpga/orca/orca4/index.cfm .
11. **Altera Co.** Stratix II Device Family Overview. – 2005. Prieiga per internetą: http://www.altera.com/products/devices/stratix2/overview/st2–overview.html .
12. **Collaborative Benchmarking Laboratory.** ISCAS'85 Benchmark Information. – 1997. Prieiga per internetą: http://www. cbl.ncsu.edu/CBL_Docs/iscas85.html.
13. **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 .
14. **Bareiša E., Jusas V., Motiejūnas K., Šeinauskas R.** Testing of FPGA Logic Cells // Elektronika ir elektrotechnika, ISSN 1392 – 1215. – 2003. – Nr.7(56). – P.37–42.
15. **Synopsys Inc.** Test Automation. – 2005. Preiga per internetą: http://www.synopsys.com/products/solutions/galaxy/test/test.html.

Pateikta spaudai 2005 03 15

**V. Abraitis, E. Bareiša. Programuojamosios logikos elemento gedimo modelis // Elektronika ir elektrotechnika. – Kaunas: Technologija, 2005. – Nr. 6(62). – P. 52–56.**

Pateikiami programuojamosios logikos elemento gedimų modeliai. Modeliai sudaryti, siekiant gauti išsamesnius testus schemoms, realizuotoms naudojant programuojamuosius lustus. Naudojant pateiktus modelius perrenkamos visos įmanomos kombinacijos schemos elementų įėjimuose ir šitaip gaunamas išsamesnis schemos testas. Į gedimų sąrašą įtraukiami tiktai neperteklinia gedimai. Atliktas eksperimentinis modelių patikrinimas naudojant Virtex šeimos programuojamuosius lustus. Remiantis gautais rezultatais galima teigti, kad pateikti modeliai naudotini testuojant programuojamuosius lustus. Siūlomi gedimų modeliai leidžia pritaikyti tradicinius testinių sekų generavimo būdus ir testavimo priemones. Il. 3, bibl. 15 (lietuvių kalba; santraukos lietuvių, anglų ir rusų k.).

**V. Abraitis, E. Bareiša. The Fault Model of Programmable Logic Block // Electronics and Electrical Engineering. – Kaunas: Technologija, 2005. – No. 6(62). – P. 52–56.**

There are presented the fault models of programmable integrated circuits in this paper, when programmable integrated circuits are configured to implement a given application. Proposed fault model can be used with traditionally automatic test sequence generators and result will be exhaustive test for programmable integrated circuits with given configuration. Model was tested using Virtex family PFGA's. Ill. 3, bibl. 15 (in Lithuanian; summaries in Lithuanian, English and Russian).

**В. Абрайтис, Э. Барейша. Моделъ неисправностей элемента программируемых интегральных схем // Электроника и электротехника. – Каунас: Технология, 2005. – № 6(62). – С. 52–56.**

Рассматриваются модели неисправностей элементов программируемых интегральных схем (ИС), когда программируемые интегральные схемы запрограммированы выполнять данную функцию. Предлагаемые модели неисправностей могут быть использованы с традиционными генераторами тестовых последовательностей и результатом будет исчерпывающий тест для программируемой интегральной схемы с данной конфигурацией. Модели были проверены, используя программируемые интегральные схемы семейства Virtex. Ил. 3, библ. 15 (на литовском языке; рефераты на литовском, английском и русском яз.).

DOI: 10.5755/j02.eie.10454