KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
PRAKTINĖS INFORMATIKOS KATEDRA

Vytautas Survila

ABSTRAKČIŲ AUTOMATŲ VALDOMUMO TYRIMO PROGRAMINĖ ĮRANGA

Magistro darbas

Vadovas
prof. habil. dr. R. Šeinauskas

KAUNAS, 2005

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIOS FAKULTETAS
PRAKTINĖS INFORMATIKOS KATEDRA

TVIRTINU
Katedros vedėjas
dr. E. Bareiša
2005–06–11

ABSTRAKČIŲ AUTOMATŲ VALDOMUMO TYRIMO PROGRAMINĖ ĮRANGA

Informatikos magistro baigiamasis darbas

Kalbos konsultantė
Lietuvių kalbos katedros lektorė
dr. J. Mikelionienė
2005–05–23

Recenzentas

2005–05–27

Vadovas
prof. habil. dr. R. Šeinauskas
2005–05–30

Atliko
IFM 9/2 gr. stud.
V. Survila
2005–05–27

KAUNAS, 2005

SUMMARY

One of possible expenditures of reduction and testability process acceleration choices is to increase circuit's testability. It means to increase controllability and observability of the circuit. As to determine the circuit's controllability and observability manually takes lots of time, it is meaningful to automate this process.

The main purpose is to determine and improve the controllability of the circuit by offering some suggestions to system on chip's designers how to improve circuit's controllability. We try to analyze if it is possible to do this only by using created software. This software's architecture uses client-server mode and all computations are performed on a server side. The system is realized on Apache server with Linux OS. System modules are realized using C++, PHP, JAVA, HTML and JavaScript programming languages.

In this paper it is being introduced the definition of controllability, explored methods of controllability estimation and increase. Also there is introduced the research how to improve accuracy of software results. The methods of the controllability estimation for "white box" are modified to work with "black box". Assumptions made during the research were validated by experiments.

TURINYS

Lentelių sąrašas:

Paveikslų sąrašas:

1. Įvadas

1.1. Temos aktualumas

Šiuolaikinių elektroninių schemų testavimas tampa brangus ir sudėtingas. Vienas iš galimų sąnaudų sumažinimo ir testavimo proceso pagreitinimo variantų yra padidinti schemų testuojamumą, t. y. padidinti schemos valdomumą ir stebimumą. Kadangi nustatyti schemos valdomumą bei stebimumą rankiniu būdu užima daug žmogaus darbo valandų, todėl yra tikslinga šį procesą automatizuoti.

Šiam tikslui įgyvendinti nuspręsta kurti programinę įrangą, kuri projektuotojams padėtų nustatyti tam tikrų schemos taškų stebimumo ir valdomumo koeficientus. Gauti duomenys pagelbės projektuotojams lengviau testuojamų schemų (*design for testability*) kūrime.

Projektavimo metu schemos testinio varianto realizavimas susijęs su didelėm papildomom išlaidom. Todėl kilo idėja schemos valdomumo ir stebimumo koeficientų nustatymui naudoti abstrakčius – imitacinius modelius, realizuotus aukšto lygio programavimo kalba. Imitaciniai modeliai – abstraktūs automatai yra programuojami pagal schemos specifikaciją. Abstraktaus automato privalumas prieš realią schemą yra jo modifikavimo paprastume. Yra pigiau modifikuoti modelio programinį kodą, negu fizines schemos dalis.

1.2. Darbo tikslas

Šio darbo pagrindinis tikslas yra nustatyti schemos valdomumą, kai nežinom jos struktūros, t. y. nustatyti automato, kuris tiriamas kaip „juoda dėžė“, valdomumą bei jį pagerinti, pateikiant schemų projektuotojams pasiūlymų, kaip būtų galima padidinti duotos schemos valdomumą.

Tai bus tyrimas, ar galima programiniu būdu pagerinti automatų valdomumą. Automatizuotas valdomumo ir valdomumo pagerinimo procesas suteiks labai daug privalumų schemų projektuotojams. Tai smarkiai sumažins schemos projektavimo laiką, be to, tai turi pagerinti schemų testuojamumą, o tai reiškia, kad sumažės schemų eksploatacinės išlaidos.

Dėl šios priežasties vienas iš tikslų yra sukurti PĮ, kuri automatiškai įvertintų ir pagerintų abstraktaus automato valdomumą.

Tyrimo metu yra nustatoma kaip galima modifikuoti programinę įrangą, kad gauti duomenys apie automato valdomumą būtų kiek galima tikslesni.

1.3. Darbo struktūra

Pirmajame skyriuje (įvade) apibrėžiamas temos aktualumas, valdomumo sąvoka, trumpai apžvelgiama taikymo sritis, kuriai skirta suskurta programinė įranga.

Antrajame skyriuje pateikiama teorinė šio projekto dalis. Yra apžvelgiami valdomumo nustatymo ir pagerinimo būdai. Egzistuojantys metodai skirti valdomumo apskaičiavimui „baltoms dėžėms“, o kadangi šio darbu metu tiriami automatai nagrinėjami kaip „juodos dėžės“ todėl panaudojus ankščiau aprašytus metodologijas buvo sukurtas naujas metodas.

Trečiajame skyriuje pristatoma sukurta programinė įranga. Trumpai aprašomos sistemos teikiamos funkcijos. Detaliau išnagrinėtos valdomumo nustatymo ir valdomumo pagerinimo realizacijos ypatybės.

Ketvirtas skyrius skirtas tyrimui. Šiame skyriuje pateikiami tiriami objektai bei šių tyrimų rezultatai. Pateikiamos išvados apie sukurtos sistemos efektyvumą bei jo pagerinimą.

Toliau pateikiamos bendrosios išvados, literatūros sąrašas, terminų bei santrumpų žodynėlis ir priedai, reikalingi anksčiau pateiktoms mintims bei faktams pagristi.

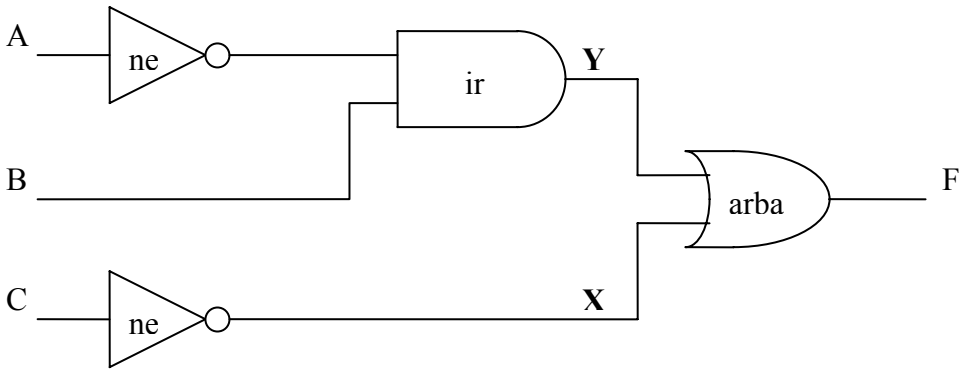
2. Teorinė dalis

Testavimo lengvumas (*testability*) yra vienas iš svarbiausių reikalavimų, į kurį reikia atsižvelgti kartu su kitais esminiais apribojimais, t. y. našumu ar kaina [6]. Sunkiai testuojama schema sukelia didelius kainos ir laiko nuostolius po pagaminimo, funkcionalumo testavimo metu [1]. Ši pusė yra labai svarbi, nes funkcionalumo testavimai atliekami visą schemos gyvavimo laikotarpį. Yra atrasta keletas gerų metodų sistemų schemos testavimui pagerinti ir visi metodai susiję su valdomumo ir stebimumo pagerinimu.

Šio projekto tikslas yra pagerinti automato valdomumą programiškai, tai yra programa vartotojui pasiūlys schemos pagerinimo variantus. Automatai bus imituojami C/C++ kalba ir visi veiksmai su jais bus atliekami kaip su „juodomis dėžėmis“ (programa neanalizuos schemų vidaus struktūros).

2.1. Automato valdomumo samprata

Valdomumas – matas, nusakantis, kaip lengva ar sunku nustatyti pasirinktą schemos mazgą į 0 arba 1 [5]. Remiantis schema, pateikta 1. pav., galima nustatyti, kad tik dvi iš aštuonių galimų įėjimų kombinacijų gali tašką Y nustatyti 1. O X nustatyti į 0 galima su puse visų galimų įėjimų kombinacijų. Iš to galima daryti išvadą, kad geriau valdomas yra X taškas.



1 pav. Tiriama schema

Pagal tiriamą schemą yra užpildoma 1 lentelė.

1 lentelė. Taško Y valdomumas

A	B	C	Valdomumas Y=1
0	0	X	–
0	1	X	✓
1	0	X	–
1	1	X	–

2 lentelė. Taško X valdomumas

A	B	C	Valdomumas X=0
X	X	0	–
X	X	1	✓

Iš pateikto pavyzdžio matyti, kad schemos valdomumas mažas. Tai reiškia, kad bus sunku nustatyti norimus schemos mazgus į norimas būsenas (1 arba 0).

2.2. Automatų imitacija

Siekiant, kad programa būtų prieinama interneto ryšį bei naršyklę turintiems vartotojams, buvo pasirinkta imituoti automatus, o ne dirbti su realiais įrenginiais. Be to, vienas iš programos tikslų – pagerinti automato valdomumą, o tai padaryti įmanoma tada, kai automatas dar nepagamintas, o tik projektuojamas, todėl geriausia atlikti bandymus jį simuliuojant.

C++ buvo pasirinkta imitavimo kalba, nes šią kalbą moka dauguma tiksliuosius mokslus studijuojančių studentų. Be to, C++ nedaug atsilieka nuo VHDL (aukšto lygmens aparatūros aprašymo kalba). Programavimo požiūriu esminis skirtumas tarp VHDL ir C++ yra lygiagrečių procesų palaikymas [9]. VHDL lygiagrečių procesų palaikymas yra labai gerai išvystytas. Be to, VHDL kalba yra pripažinta automatų projektuotojų ir šiai kalbai yra sukurta įvairių pagalbinių priemonių, kurios labai palengvina schemų projektuotojų darbus.

Tačiau esminis C++ pranašumas prieš VHDL – populiarumas [14].

2.3. Valdomumo apskaičiavimo metodai

Yra daug populiarių valdomumo apskaičiavimo metodų ir matavimo vienetų.

2.3.1. HLTS

Valdomumas ir stebimumas schemoje yra priskiriamas taškams [7].

Schemos valdomumą nusakantys faktoriai:

- CC – yra tarp 0 ir 1, kur 1 yra geriausias valdomumas
- SC – nusako kiek reikia laikrodžio impulsų, kad valdyti tašką

Yra sukurta įvairių euristikų, kaip surasti komponento valdomumo perdavimo faktorių CTF (*Controllability Transfer Factor*) [12]. CTF nusako tikimybę nustatyti sekcijos išėjime norimą reikšmę atsitiktinai paduodant įėjimus.

Kombinacinio komponento valdomumas:

$CC_{out} = CC_{in} * CTF_U$

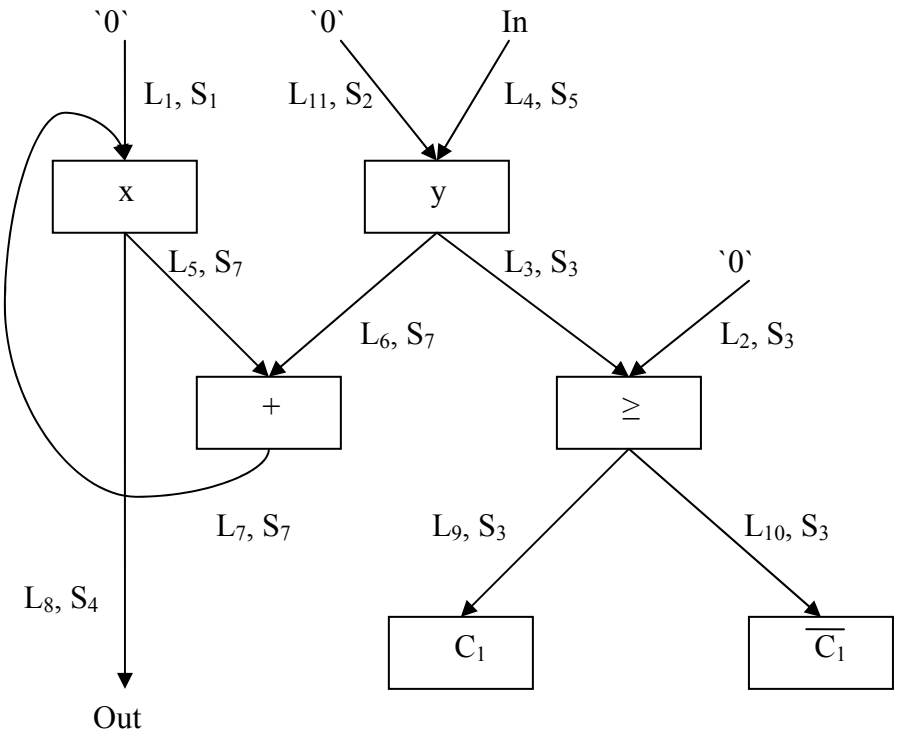
$SC_{out} = SC_{in} + clk\ (\ S_i\ )$

clk (S_i) – yra laikrodžio impulsų kiekis, reikalingas atlikti operacijai šiame funkciniame vienete.

Valdomumo skaičiavimo algoritmas [7]:

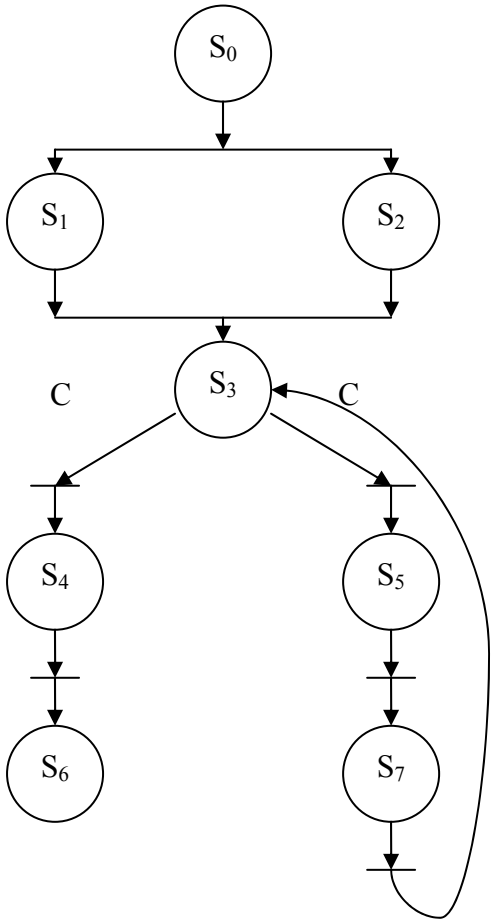
```
for visiems pirminiams įėjimams do
        CC_pirminis_įėjimas = 1;
        SC_pirminis_įėjimas = 0;
repeat
        išrinkti sekantį funkcinį vienetą U;
        // suskaičiuoti valdomumą funkcinio vieneto U
        // išėjimuose, pagal vidutinį valdomumą suskaičiuotą
        // jo įėjimuose
        if U yra kombinacinis vienetas then
                CC_išėjimas = CC_įėjimas * CTF_U;
                SC_išėjimas = SC_įėjimas + clk(S_i);
        if U yra nuoseklusis vienetas then
                CC_išėjimas = CC_įėjimas * CC_sąl(Si, Sj);
                SC_išėjimas = SC_įėjimas – clk(S_i, S_j);
        if U yra įtrauktas į grįžtamojo ryšio ciklą then
                CC_išėjimas = CC_išėjimas * CC_ciklas;
                SC_išėjimas = SC_išėjimas + SC_ciklas;
until visi pirminiai išėjimai pasiekti;
```

Pavyzdys:



2 pav.Testuojama schema

Kad būtų patogiau dirbti, schemą reikia suskaidyti į funkcinius vienetus (3 pav.).



3 pav.Schema suskaidyta į funkcinius vienetus

Atlikti skaičiavimus ir užpildyti lentelę galima tada, kai schema suskaidoma į funkcinius vienetus (3 lentelė).

3 lentelė. Schemos testavimo faktoriai

Taškas	CC	SC
L_1	–	–
L_2	–	–
L_3	1,00	2
L_4	1,00	0
L_5	0,62	6
L_6	1,00	1
L_7	0,72	3
L_8	0,62	5
L_9	0,86	2
L_{10}	0,86	2
L_{11}	–	–

Atlikus skaičiavimus galime daryti išvadą, kad lengviausiai valdomos jungtys yra L_3, L_4 ir L_6. Sunkiausiai valdomos – L_5 ir L_8. Iš to galime spręsti, kad atsitiktinai generuojant įėjimus, bus sunkiausia pakeisti jungčių L_5 ir L_8 reikšmes.

2.3.2. TMEAS

Šio metodo skaičiavimo vienetai yra normalizuojami tarp 0 ir 1, tam kad atspindėti schemos taško valdymo lengvumą. Metodo žingsniai [13]:

- Kiekvienam schemos taškui s valdomumą pažymim kaip CY(s).
- CY(s) reikšmės yra gaunamos sprendžiant lygtis, tuo pačiu laikant kad CY(s) yra nežinomi:
    - Jei įėjimus pažymėsime x_1, x_2, ..., x_n, o išėjimus z_1, z_2, ..., z_m tada CY kiekvienam išėjimui z_j bus:

$$CY(z_j) = CTF \times \frac{1}{n}\sum_{i=1}^{n} CY(x_i),$$

    - CTF – valdomumo perdavimo faktorius. Jei $N_j(0)$ ir $N_j(1)$ yra įėjimų kombinacijų skaičius, kuriam z_j atitinkamai turi reikšmę 0 ir 1 tada:

$$CTF \equiv \frac{1}{m}\sum_{j=1}^{m}\left(1 - \frac{\left|N_j(0) - N_j(1)\right|}{2^n}\right)$$

- Jei s dalinasi į k šakų, tada CY kiekvienam išsiskleidžiančiam taškui yra:

$$CY = \frac{CY(s)}{(1+\log k)}$$

2.3.3. SCOAP

Sandia valdomumo ir stebimumo analizės programa (*the Sandia Controllability Observability Analysis Program*). Tai vienas iš labiausiai naudojamų būdų valdomumui ir stebimumui nustatyti[15]. Skaičiavimo vienetai atspindi vidinių schemos taškų valdymo sunkumą, tad kuo skaičius didesnis tuo sunkiau juos valdyti [13].

Pradžioje išnagrinėsime trijų įėjimų IR elementą, kai Y=IR(A, B, C). Tada Y valdomumas vienetu ir nuliu yra:

$CC^1(Y) = CC^1(A) + CC^1(B) + CC^1(C) + 1$

$CC^0(Y) = \min (CC^0(A), CC^0(B), CC^0(C)) + 1$

Rezultatas yra didinamas vienetu tam, kad iš dalies atspindėtų atstumą nuo pirminių įėjimų. CC kiekvienam schemos taškui galima suskaičiuoti einant nuo pirminių įėjimų į pirminius išėjimus.

Nuoseklusis valdomumas (*sequential controllability*) parodo apytikslį laikrodžio impulsų kiekį, reikalingą nustatyti tam tikrą tašką į tam tikrą būseną [1]. Skaičiuojant nuoseklųjį valdomumą kombinacinėms schemoms reikšmės nėra didinamos vienetu – nereikia papildomų laikrodžio impulsų. O skaičiuojant nuosekliosioms schemoms SC yra didinama vienetu, o CC nedidinama.

2.3.4. CAMELOT

Kompiuteris – schemų testuojamumo įvertinimo pagalbininkas (*the Computer – Aided MEasure of LOgic Testability*) skirtas, kad pagerinti TMEAS [4]. Valdomumo reikšmės CY gali kisti nuo 0 iki 1. Gautos reikšmės atspindi taškų valdomumo lengvumą. CY bus skaičiuojamas pagal:

CY(išėjimas) = CTF(išėjimas) * f(CY(įėjimo))

kur CTF – komponento valdomumo perdavimo faktorius, o funkcija f apjungia visus CY įėjimus, kurie yra susiję su išėjimu. CTF skaičiavimas [13, 14]:

$$CTF = 1 - \left| \frac{N(0) - N(1)}{N(0) + N(1)} \right|,$$

2.4. Valdomumo padidinimo metodai

Egzistuoja įvairūs automatų valdomumo pagerinimo metodai. Juos galima sugrupuoti į dvi kategorijas: specifinius ir struktūrizuotus metodus [2].

2.4.1. Specifiniai metodai

Egzistuoja daug specifinių schemų testavimo metodų. Vienas iš jų veikia vidinius signalus multipleksuojant į išorines suderinimo jungtis[2]. Tada šie signalai, kurie bus naudojami automatui testuoti, gali būti išgaunami naudojant tam tikrus schemų analizatorius. Kiti metodai sudaryti iš savaiminio testavimo (*self – test*) pvz., tokia yra parašo analizė [3]. Ši

technika naudoja amžino ciklo nustatymą, kad patikrintų įvairių testinių sekų rezultatus. Deja, šie du metodai yra sunkiai pritaikomi ir siūlo tik ribotą arba visiškai nesuteikia informacijos apie tikrąją automato būseną.

Gana paplitęs būdas yra naudoti schemų analizatorius, kurie yra įmontuoti į schemas [16]. Tai suteikia galimybę realiu laiku pasiekti bet kurį tašką luste, taip pat ir galimybę keisti trigerių būsenas. Nors bet kuriuo metu pasiekiamų signalų kiekis yra didesnis nei išorinio schemų analizatoriaus, bet tas kiekis turi suvaržymų ir valdantys signalai turi būti nustatyti prieš testo pradžią. Norint stebėti daugiau ar kitus signalus arba daryti kokius nors kitus pakeitimus įmontuotame schemos analizatoriuje reikia visiško schemos konstrukcijos perrinkimo, kadangi tai paveikia lusto konfigūraciją ir dydį.

Šių specifinių metodų privalumai yra tie, kad jie beveik nepakeičia schemos ploto ir greičio, ir juos naudojant galima vykdyti testus, kai schema dirba realiu greičiu [11]. Jie labai naudingi greičio testavimui sunkiai prieinamose schemos vietose. Deja, specifiniai metodai teikia ribotas galimybes pilnam funkciniam verifikavimui. Visų pirma, dažniausiai jie yra specifiniai kiekvienam projektui – jie reikalauja projektuotojo įsikišimo, kad įterpti ir naudoti [10]. Antra, iš vartotojo jie reikalauja nurodyti norimus stebėti signalus prieš realizuojant schemą ir, norint atlikti kokius nors pakeitimus, reikia schemą perrinkti iš naujo. Visa tai reikalauja daug laiko ir didelio bitų kiekio apdorojimo. Trečia, jie suteikia ribotą schemos būsenų matomumą– matomi tik signalai nukreipti į įmontuotą schemos analizatorių yra. Galiausiai, jie nesuteikia galimybės keisti schemos būsenas.

2.4.2. Struktūrizuoti metodai

Žinomas struktūrizuoto metodo pavyzdys yra pakartotinio skaitymo metodas (*readback*). Siekiant jį pritaikyti naudojamas įmontuojamas mechanizmas, kuris leidžia vartotojui išgauti schemos konfigūracijos bitų srautą [11]. Konfigūracijos bitų srautas pats vienas neduoda jokios naudos, bet kai išskaitomas iš schemos, tai jį sudaro esamos schemos būsenos trigerių ir atminčių reikšmės. Tada ši būsena gali būti užkrauta į simuliatorių, kur bus galima pamatyti visas tikrąsias sistemos būsenas.

Kitas metodas naudoja dalinį perkonfigūravimą, kad užkrauti vidines schemos būsenas iš išorinio šaltinio, siekiant nustatyti schemą į norimą būseną, vykdant funkcinį verifikavimą [17]. Tai leidžia vartotojams ištestuoti ribinius atvejus arba nustatyti schemą į žinomą būseną po kurios įvyksta klaida. Jei negalėtumėme iš išorės reguliuoti būsenų, tada reiktų ilgai įėjimų kombinacijas padavinėti schemai, kol nustatytume schemą į norimą būseną. Kartais yra neįmanoma parinkti tokios įėjimų sekos, kad nustatyti schemą į norimą būseną.

Trečias metodas naudoja skanavimo rinkinius (*scan chain*), kurie yra įterpiami į schemą. Ši skanavimo versija leidžia informacijai, esančiai trigeriuose ir įmontuotose atmintyse būti nuosekliai išgaunamai per papildomą jungtį [19].

Struktūrizuoti metodai turi daug privalumų. Pirma, struktūrizuoti metodai pajėgūs suteikti pilną schemos stebimumą ir valdomumą funkciniam verifikavimui [4]. Taigi, ne tik visi schemos signalai yra matomi, bet yra ir galimybė nustatyti schemą į norimą būseną, kad pagerinti verifikavimą. Specializuoti metodai leidžia projektuotojui stebėti tik dalį schemos, kai ji dirba pilnu greičiu, ir nesuteikia galimybės keisti schemos būsenų. Antra, kadangi struktūrizuoti metodai suteikia galimybę stebėti visas schemos būsenas, tada schemai užtenka vienos realizacijos, sumažinant laiko ir medžiagų sąnaudas, kurios reikalingos taikant specifinius metodus. Trečia, metodai, tokie kaip skanavimas gali būti mechanizuoti, nes jie nepriklauso nuo konstrukcijos, o tokį procesą galima automatizuoti.

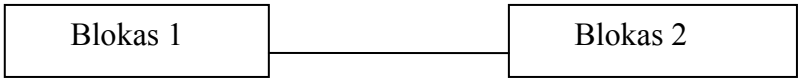
Visgi egzistuoja ir keletas blogų struktūrinių metodų pusių [4]. Pavyzdžiui, vienas iš pakartotinio skaitymo nepatogumų yra tas, jog reikia sustabdyti laikrodį, kad atlikti pažingsninį testavimą. Tai vyksta dėl to, kad skiriasi pakartotinio skaitymo mechanizmas, naudojamas trigeriams ir įmontuotoms atmintims. Kad palaikyti nuoseklumą tarp trigerių ir atminčių, laikrodis turi būti sustabdytas tęsiantis skaitymui. Be to, skanavimas žymiai padidina schemos ploto ir greičio sąnaudas.

Kadangi dalinio perkonfigūravimo metodas yra lengvai suprantamas ir pritaikomas – jis yra išnagrinėtas detaliau.

Dalinio perkonfigūravimo metodas nagrinėjama šias metodologijas [8, 10]:

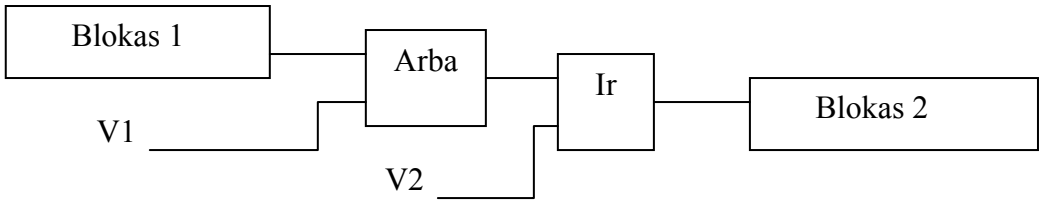
- Testavimo taškų metodas;
- Testavimo taškų multipleksavimas ir demultipleksavimas;
- Įėjimų ir išėjimų pasidalinimas normaliam ir testavimo darbo režimams;

Testavimo taškų metodas:



4 pav.Testavimo taškų metodas

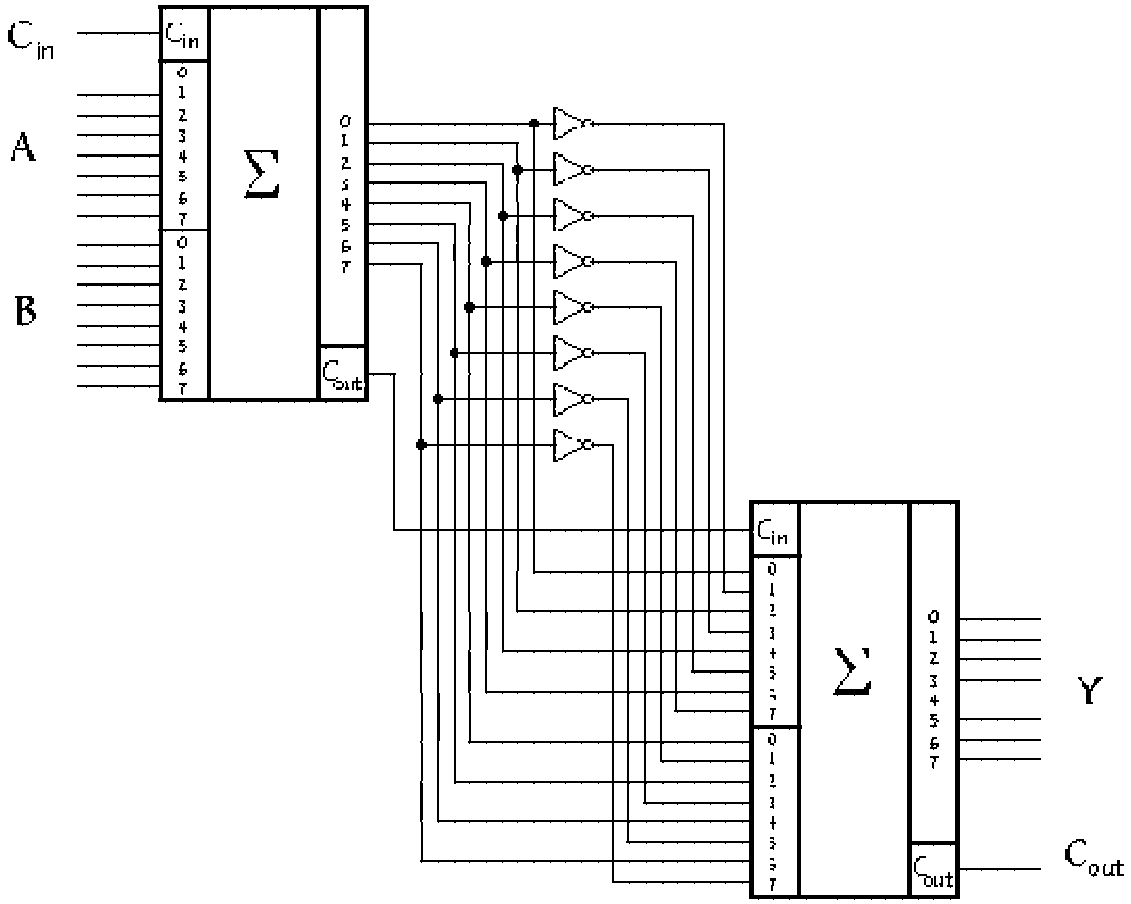
Blokas 2 yra nevaldomas. Kad pagerinti valdomumą reikia Bloką 2 padaryti valdomą iš išorės (5 pav.):



5 pav.Testavimo taškų metodas su pagerintu valdomumu

Kai V1 = 0, o V2 = 1 tada schema dirba normaliu darbo režimu. Jei norime į Bloką 2 paduoti 0, tai yra valdyti Bloką 2 nuliu, tada V2 turi būti lygus 0. O jei norim Blokas 2 valdyti vienetu, tada V1 ir V2 turi būti lygus 1.

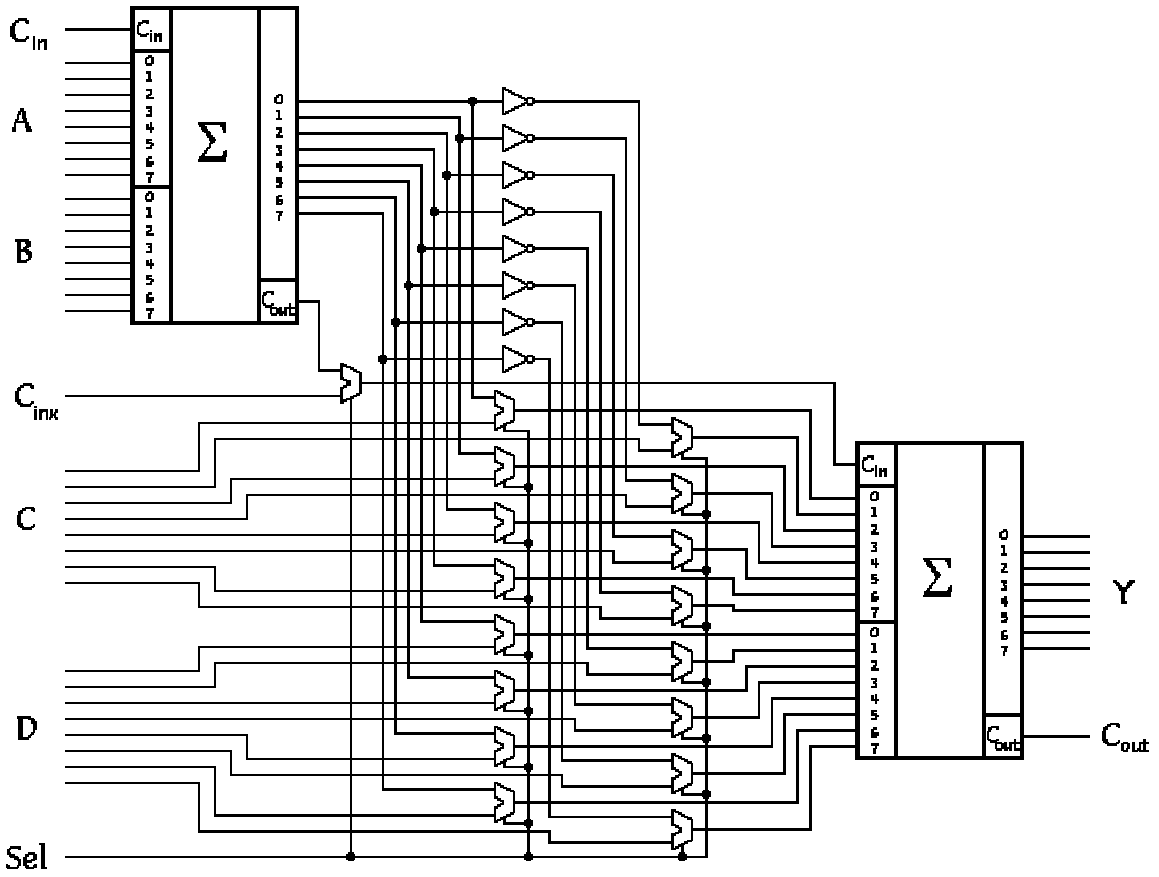
Realus pavyzdys papildomų taškų pridėjimui pateikiamas 6 pav. [6]:



6 pav.Originali schema

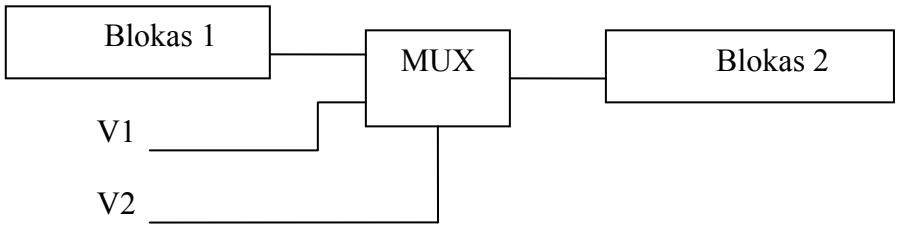
Šiame pavyzdyje nagrinėjami nuosekliai sujungti du aštuonių bitų sumatoriai, tai yra pirmo sumatoriaus išėjimai sudėtingu būdu yra prijungti prie antro sumatoriaus įėjimų. Sujungime yra dubliavimas ir tai blogai veikia antro sumatoriaus valdomumą. Taigi, tai yra sunkiai testuojamas įrenginys, jeigu nebus įvesta pataisymų testavimui pagerinti.

Kad padidinti šios schemos valdomumą, reikia prie jungiančių pirmą ir antrą sumatorius jungčių pridėti devynis įėjimus (C_{inx}, C (c_0 – c_7), D (d_0 – d_7). Taip pat reikia pridėti vieną įėjimą Sel, kuris nustatys schemos darbo režimą – normalų arba testavimo (7 pav.).



7 pav.Valdoma schema

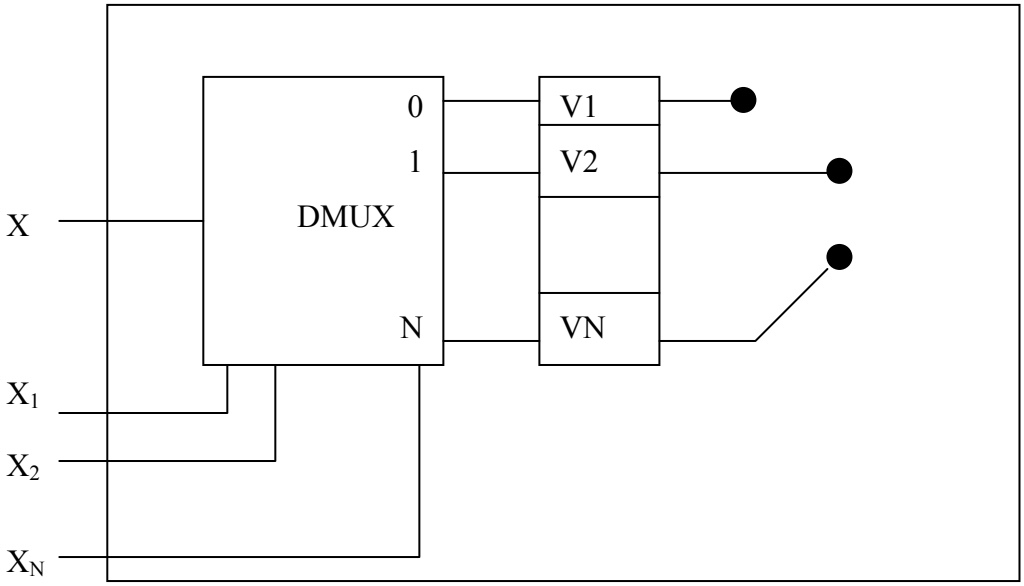
Vietoj elementų „Arba“ ir „Ir“ galime įdėti multiplekserį [4]. Tada testavimo taškų schema (5.pav.) atrodytų taip:



8 pav.Testavimo taškų metodas su multiplekseriu.

Normalus programos režimas yra kai V2 = 0. Tada Blokas 2 bus valdomas 0 kai V1 = 0, o V2 = 1. Blokas 2 valdomas vienetu, kai V1 = 1, V2 = 1.

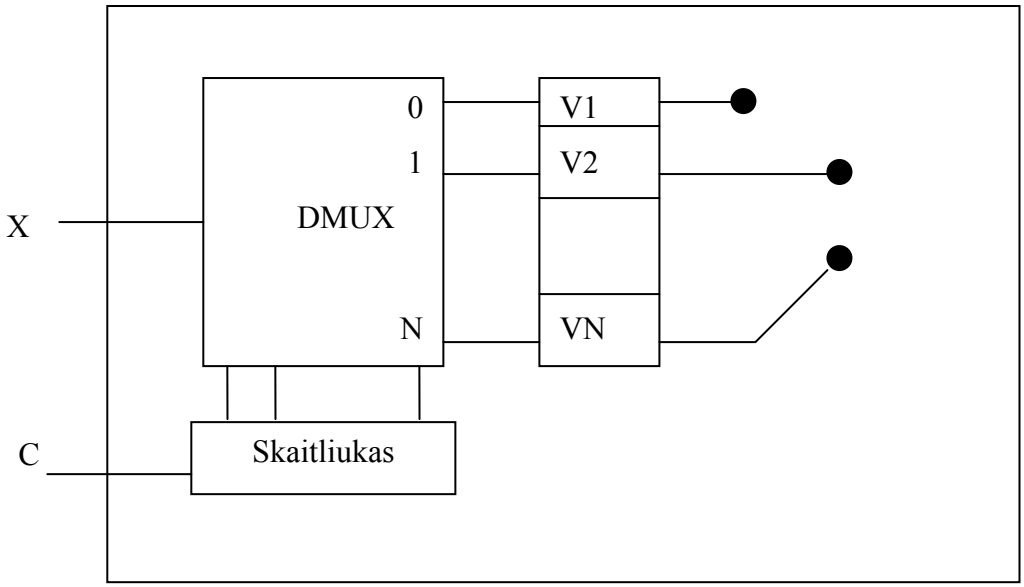
Kad sumažinti įėjimų kiekį reikalingą testinių taškų valdymui, galima naudoti demultiplekserį (9 pav.) ir nustatantį registrą:



9 pav.Įėjimų sumažinimas su demultiplekseriu.

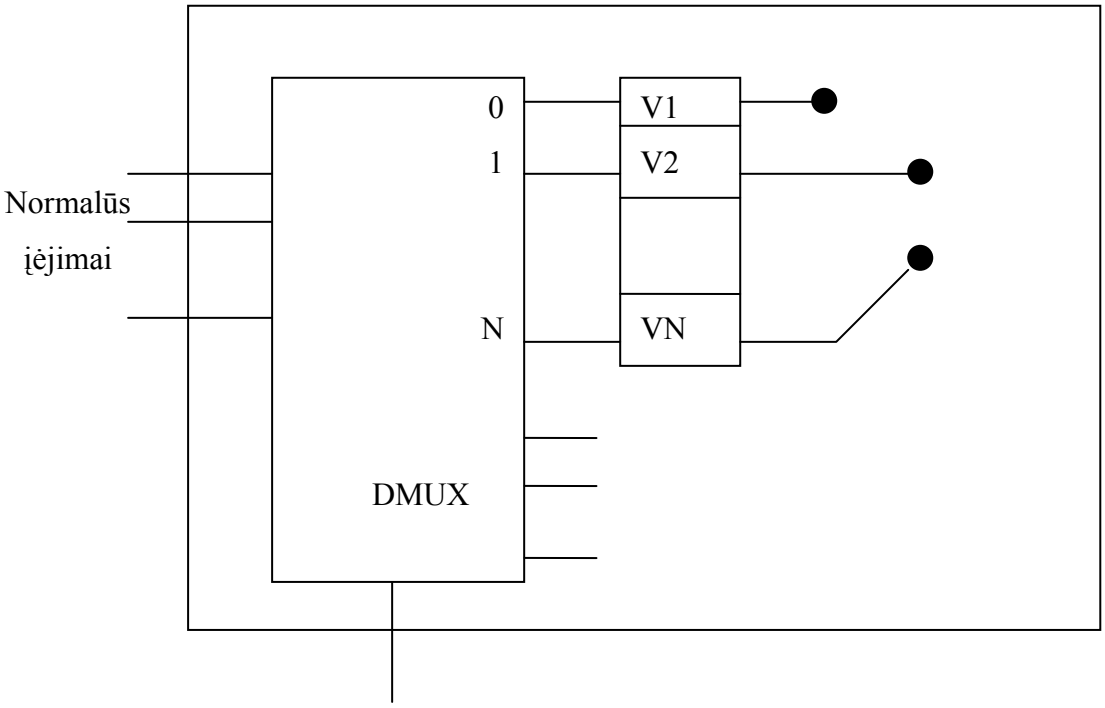
Kad nustatyti visas valdomas reikšmes, reikia N laikrodžio impulsų.

Jei norime sumažinti adresavimo ($X_1 - X_N$) įėjimus, tada reikia naudoti skaitliuką (arba postūmio registrą):



10 pav. Įėjimų sumažinimas su demultiplekseriumi ir skaitliuku.

Kad sumažinti įėjimų, skirtų testinių taškų valdymui skaičių, taip pat galima naudoti įėjimų pasidalinimą tarp normalaus ir testavimo darbo režimų:



11 pav. Įėjimų pasidalijimas tarp normalaus ir testavimo režimų.

Naudojant pastarąjį metodą, schemoje prisideda tik vienas papildomas įėjimas. Jis nustato, ar duomenys paduodami į įėjimus yra siunčiami testinių taškų valdomumui, ar perduodami į tikruosius įėjimus. Jei testinių taškų, kuriuos norime kontroliuoti, yra daugiau nei originalių įėjimų, tada reikia dviejų ar daugiau valdančiųjų įėjimų.

2.5. Metodo pasirinkimo problemos

Aukščiau pateikiamuose metoduose automatai yra nagrinėjami kaip „baltos dėžės“, t. y. kai yra žinoma jų vidinė struktūra. Kadangi šios sistemos tikslas yra ištirti valdomumą automatų, kurie yra pateikiami kaip „juodos dėžės“, buvo nuspręsta sukurti naują metodą, kuris, remdamasis „baltų dėžių“ valdomumo nustatymo principais, apskaičiuotų valdomumą automatams, kurie aprašyti kaip „juodos dėžės“.

Pilnas visų galimų įėjimų perrinkimo metodas nenaudojamas, nes tiriami automatai gali turėti gana didelį įėjimų ir išėjimų skaičių, kas padarytų skaičiavimų procesą neleistinai ilgą (netikslinga laukti keletą mėnesių, kad sužinoti schemos valdomumą). Nuspręsta naudoti atsitiktinį įėjimų generavimą. Šiuo būdu yra sutaupoma programos darbo laiko, tačiau rezultatai turi paklaidą, kuri priklauso nuo generuojamo įėjimų skaičiaus.

Schemoms su atmintimi imituoti buvo pasirinktas iteracinis modelis. Schemos vidinės būsenos yra „išvedamos“ prie įėjimų ir išėjimų t. y. pradinė būsena yra nustatoma per papildomą įėjimą, o būsenos rezultatas yra išvedamas per papildomą išėjimą. Taip galima schemą su atmintimi imituoti kombinacine schema. Kad imituoti daugiau nei vieną iteraciją, reikia iš pirmos iteracijos gautas būsenas perduoti į antrą ir sekančias.

Valdomumas yra skaičiuojamas kiekvienam išėjimui ir būsenai. Reikšmės yra normalizuojamos tarp 0 ir 1. Nulis reiškia kad mazgas yra nevaldomas, o 1, kad labai gerai valdomas. Bendras automato valdomumas yra gaunamas išvedus visų išėjimų ir būsenų valdomumo vidurkį.

Kad šis metodas pateiktų tikslesnius rezultatus su vienu iteraciniu modeliu reikia atlikti keliolika ar daugiau bandymų. Galutinis rezultatas gaunamas išvedus visų bandymų vidurkį.

3. Projektinė dalis

3.1. Sistemos paskirtis

Efektyvios mikroelektronikos technologijos tyrimo priemonės bei projektavimo automatizavimo įrankiai įgalina inžinierius projektuoti vis didesnes, sudėtingesnes integruotas schemas. Sistemos lustuose yra nauja kryptis puslaidininkių pramonėje. Naujų produktų projektavimas ir testavimas yra labai brangus. Todėl sudėtingų elektroninių sistemų testavimas ir verifikavimas yra labai svarbus. Projektavimas ir testavimas yra vientisas procesas.

Testavimo lengvumas yra vienas iš svarbiausių reikalavimų, į kurį reikia atsižvelgti kartu su kitais tokiais esminiais apribojimais, kaip našumas ar kaina [3]. Sunkiai testuojama schema sukelia didelius kainos ir laiko nuostolius po pagaminimo, funkcionalumo testavimo metu. Šis aspektas yra labai svarbus, nes funkcionalumo testavimai atliekami visą schemos gyvavimo laikotarpį. Yra atrasta keletas rimtų metodų sistemų schemos testavimui pagerinti ir jie visi susiję su valdomumo ir stebimumo pagerinimu.

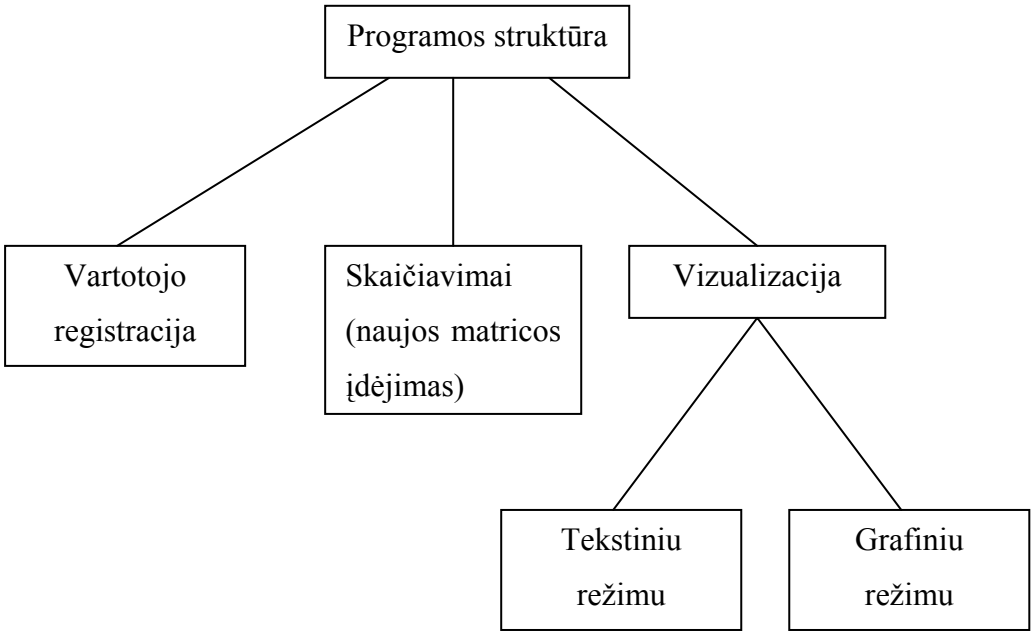
Sistemos paskirtis yra nustatyti schemos valdomumą bei pateikti pasiūlymus kaip būtų galima jį pagerinti. Valdomumas yra pateikiamas kiekvienam schemos išėjimui bei vidinei būsenai. Taip pat yra pateikiamas kiekvieno išėjimo ir būsenos valdomumas kiekvienai iteracinio modelio kopijai, t. y. galima nustatyti po kelių iteracijų galima gauti atitinkamą schemos taško valdomumą. Kaip bendras automato valdomumas yra pateikiamas ir visos schemos valdomumas, kuris yra apskaičiuojamas remiantis visų išėjimų ir būsenų valdomumo reikšmėmis.

Iš sistemos norima, kad vartotojas, turėdamas automato modelį, parašytą C/C++ kalba, iš bet kurios pasaulio vietos neturėdamas jokių papildomu darbo instrumentų, panaudojus tik internetinę naršyklę, galėtų patalpinti turimą automatą į serverį, ištestuoti jį ir gauti jo skaitines ir grafines charakteristikas, bei gauti pasiūlymų, kaip modifikuoti schemą, kad jos valdomumas padidėtų. Taip pat peržiūrėti ir susipažinti su jau esamais abstrakčiais automatais, kurie yra saugomi abstrakčių automatų archyve.

3.2. Bendra sistemos architektūra

Kadangi sistema yra gana didelė ir sudėtinga, todėl yra tikslinga ją suskaidyti atskiromis dalimis. Sistemos darbo režimus realizuoja atskiros programos dalys:

1) vartotojo registracija,
2) skaičiavimai,
3) vizualizacija.



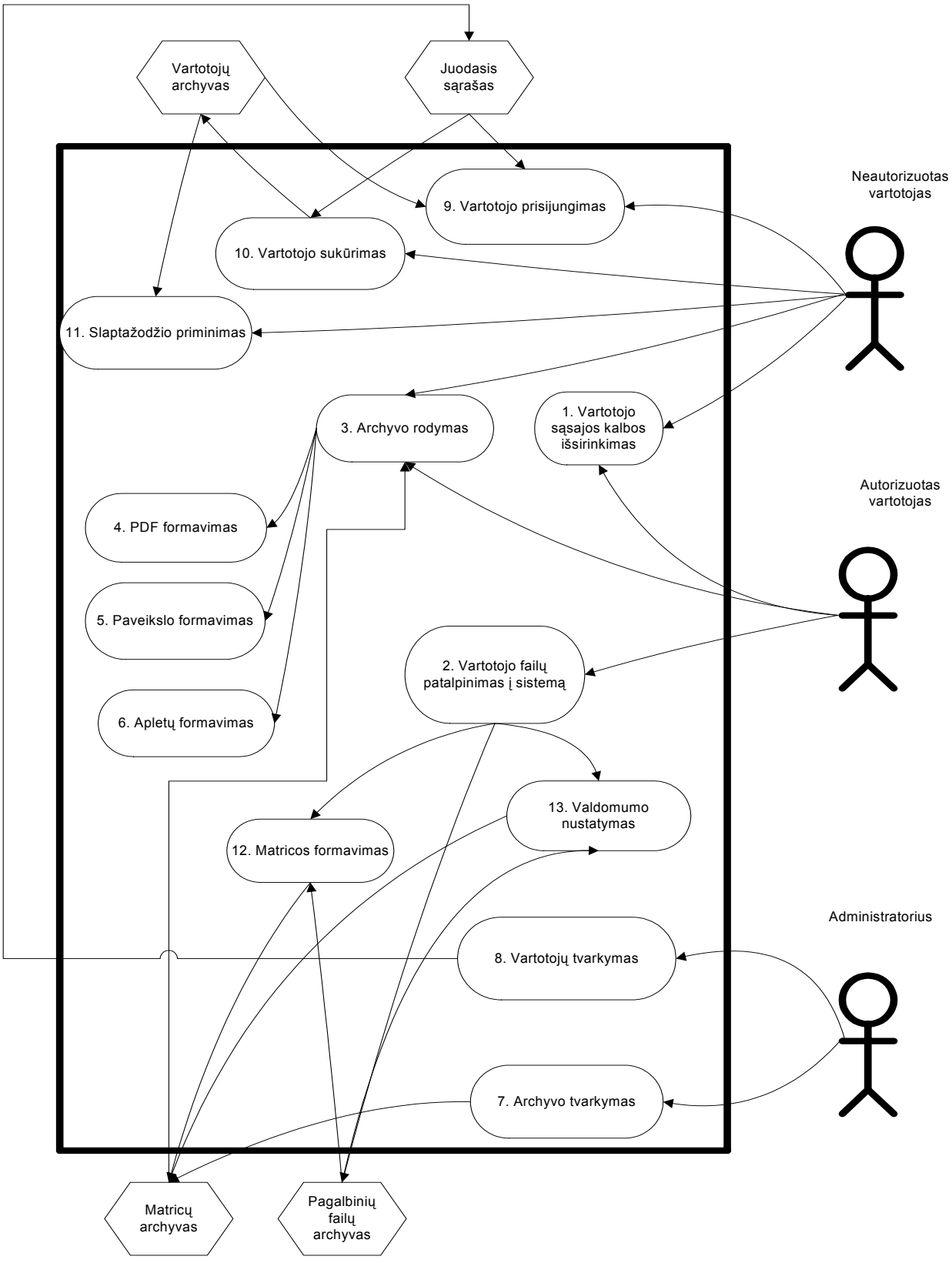
12 pav. Sistemos struktūra

Vartotojo registracijos modulis skirtas realizuoti visoms funkcijoms, kurios yra skirtos vartotojo registracijai bei autentifikavimui. Taip pat viena iš funkcijų yra slaptažodžio priminimas, bei „juodojo“ sąrašo valdymas (neleidžiama prisijungti vartotojams, kurie yra atlikę neleistiną veiksmą sistemoje).

Skaičiavimų modulis skirtas naujo automato įkėlimo funkcijoms ir skaičiavimams su sėkmingai įkeltu automatu atlikti. Šis modulis atsakingas už valdomumo apskaičiavimą ir pagerinimą.

Vizualizacijos modulis skirtas įvairiems rezultatų atvaizdavimo būdams. Galimas tekstinis atvaizdavimas, naudojamas tada kai reikia žinoti tikslias rezultatų reikšmes. Grafinis režimas naudojamas tada kai norima susidaryti bendrą nuomonę apie schemos valdomumą.

Detali sistemos architektūra pateikiama 13 pav.



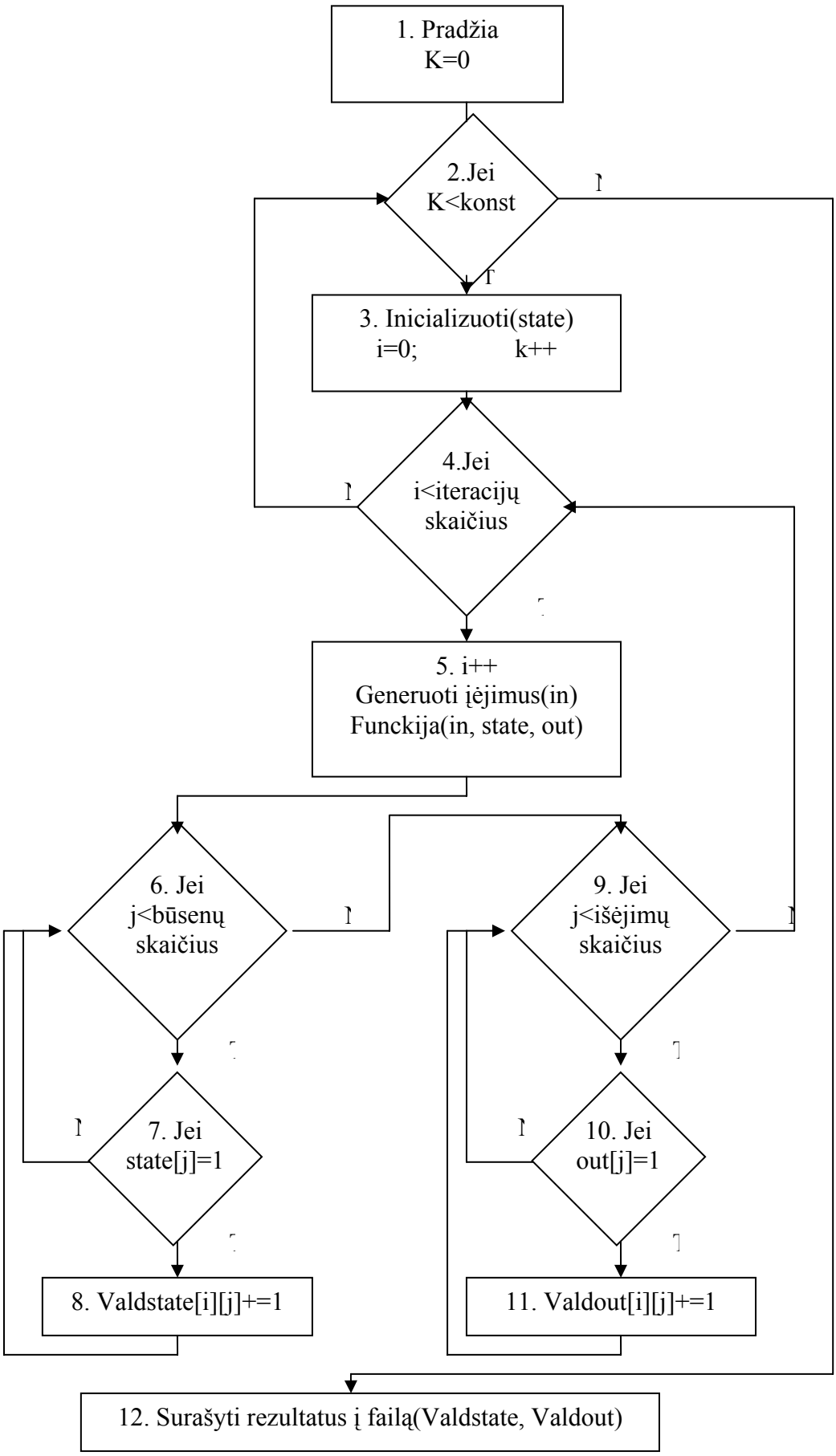
13 pav. Sistemos architektūra

3.3. Valdomumo nustatymo detali architektūra

Ši funkcija skaičiuoja abstraktaus automato išėjimų ir vidinių būsenų valdomumą. Ji yra atsakinga už korektišką abstraktaus automato išėjimų ir būsenų valdomumo apskaičiavimą. Kaip rezultatas yra pateikiamas valdomumas kiekvienam išėjimui ir būsenai po kiekvienos iteracijos.

Ši funkcija nenaudoja kitų funkcijų. Šią funkciją naudoja *Valdomumo pagerinimo* funkcija. Ji gali sukelti šalutinį poveikį jei duos nekorektiškus rezultatus (nurodys didesnį valdomumą negu yra) ir tada *ValdomumoPagerinimas* „galvos“ kad jau pagerino valdomumą ir nebebandys jo gerinti.

Skaičiavimo algoritmas pateikiamas 14 pav. „Valdomumo skaičiavimo algoritmas“.



14 pav. Valdomumo skaičiavimo algoritmas

Automato valdomumo koeficientų tikslumas priklauso nuo pasirinkto iteracijų skaičiaus, t. y., kiek kopijų bus iteraciniame modelyje. Kuo daugiau kopijų, tuo tikslesnis rezultatas.

Valdomumo skaičiavimo schema pateikta 2 pav. Aprašyme yra naudojami skaičiai skliaustuose, kurie reiškia schemoje pateikto punkto numerį.

Į pirmą iteracinio modelio kopiją yra paduodamos nulinės būsenos (automato inicializavimas (3)). Tada generuojami įėjimai (5) ir paduodami į pirmą kopiją (5). Žiūrima kur gautose būsenose yra vienetai (7) ir atitinkamai prisumuojam prie pirmos iteracijos būsenų valdomumo masyvo (8). Taip pat ieškom vienetų išėjimuose (10) ir jei randam, tai atitinkamo išėjimo valdomumą padidinam vienetu (11). Taip suskaičiuojam visom iteracijom (4) (kiekvienai iteracijai generuodami naujas įėjimų sekas, bet būsenas perduodant iš prieš tai gautų kopijų). Taip yra imituojamas kombinacinės schemos darbas ir galima stebėti jos vidines būsenas. Kai atliekami skaičiavimai, tada kiekviena gauta valdomumo reikšmė yra keičiama taip, kad ji kistų nuo 0 iki 1 (0 – nevaldomas mazgas; 1 – labai lengvai valdomas mazgas). Reikšmės yra statomos į formulę:

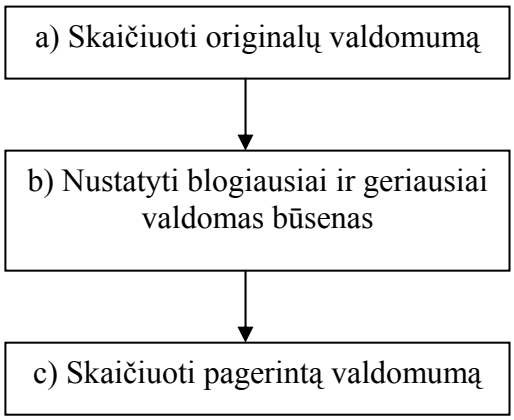
$$Nauja\ \mathrm{Re}\,ikšmė = 1 - \left| \frac{Sena\ \mathrm{Re}\,ikšmė - SkirtingųĮėjimųSkaičius}{SkirtingųĮėjimųSkaičius} \right|$$

kur: SenaReikšmė yra vienetukų skaičius, gautas pridedant po vieną, kai atitinkame išėjime yra 1 (8) arba atitinkamoje būsenoje yra 1 (11). SkirtingųĮėjimųSkaičius – kiek kartų buvo paduota sugeneruotos įėjimų sekos į skirtingus iteracinius modelius (5).

3.4. Valdomumo pagerinimo detali architektūra

Ši funkcija nustato kaip pagerinti abstraktaus automato valdomumą ir yra atsakinga už automato valdomumo padidinimą, remiantis jau apskaičiuotu automato valdomumu.

Pagerinto valdomumo skaičiavimo veiksmų seka pateikiama 15 pav.:

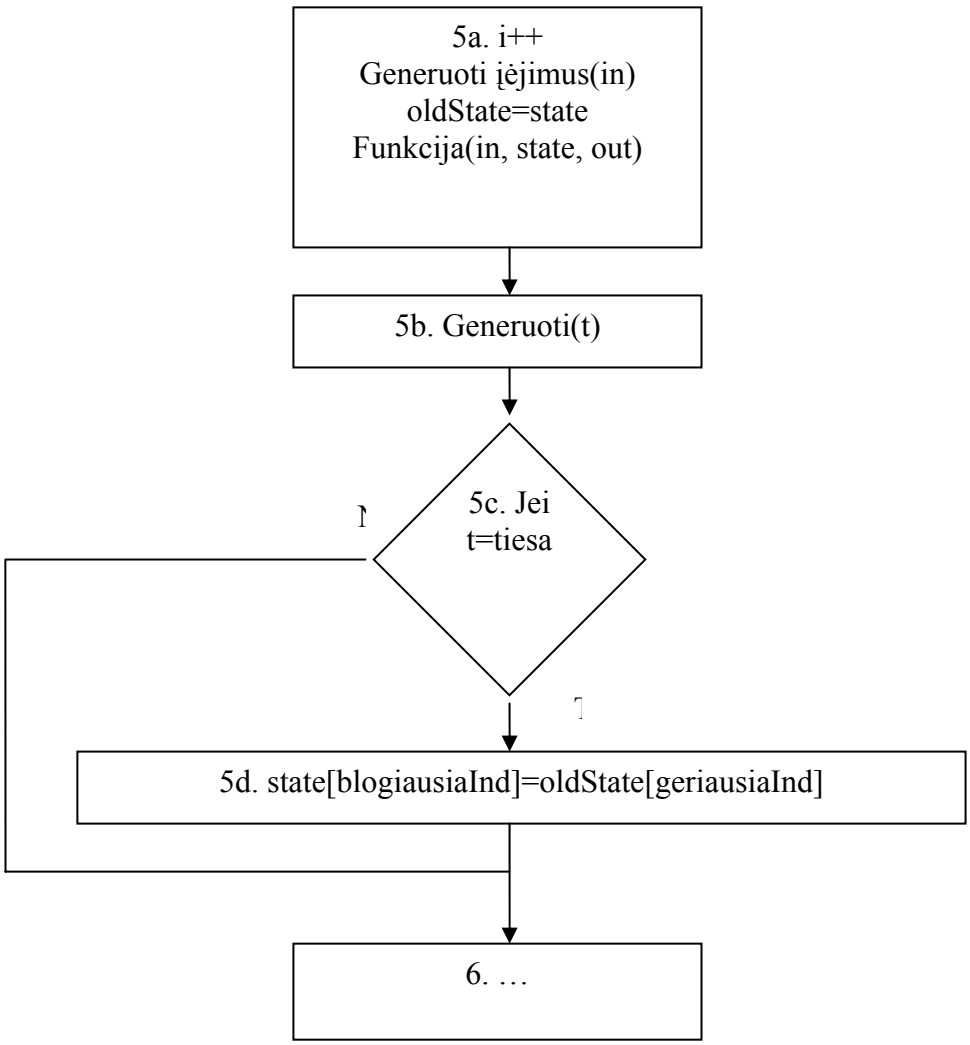


15 pav. Valdomumo pagerinimas

Valdomumo pagerinimas vykdomas atsižvelgiant į išėjimų valdomumą. Pirmiausiai yra apskaičiuojamas valdomumas originaliam automatui (a), t. y. kol dar neatlikti jokie pakeitimai. Valdomumas skaičiuojamas naudojant *Valdomumo nustatymas* funkciją. Iš gautų rezultatų vedamas vidurkis kiekvienai būsenai ir kiekvienam išėjimui, t. y. sumuojama vienos būsenos visų iteracijų valdomumo koeficientai ir gauta suma dalinama iš iteracijų skaičiaus. Tai atlikus su visais išėjimais ir visomis būsenomis išrenkamos dvi būsenos su mažiausiu ir didžiausiu valdomumu (b). Labiausiai valdoma būsena jungiama su multiplekseriu prie blogiausiai valdomos būsenos (pagerinamas blogiausiai valdomos būsenos valdomumas). Kad valdyti multiplekserių reikia pridėti papildomą įėjimą *t*, kuris nurodys ar schema dirba testiniame režime ar ne.

Kai nustatoma, kurias būsenas sujungti, – skaičiuojamas valdomumas modifikuotai schemai (c). Skaičiavimas yra toks pats kaip ir originaliai schemai jei schema dirba ne testiniam režime. Jei dirba testiniam režime, tada reikšmė gauta blogiausiai valdomoje būsenoje, yra pakeičiama reikšme, gauta geriausiai valdomoje būsenoje, sujungiamos geriausiai valdoma su blogiausiai valdoma būsenos. Rezultatų faile yra išsaugoma nekeisto automato valdomumas ir automato su padidintu valdomumu valdomumas. Į failą taip pat įrašoma būsenų numeriai, kurios turi būti sujungtos, kad padidėtų valdomumas.

Schemos su padidintu valdomumu skaičiavimo schema (pateikiama tik kaip reikia pakeisti *Valdomumo nustatymas*, schemos 14.pav., 5 punktą):



16 pav. Pagerinto valdomumo skaičiavimas

blogiausiaInd yra blogiausiai valdomos būsenos indeksas, o *geriausiaInd* geriausiai valdomos būsenos indeksas. *t* yra schemos darbo režimas: Jei tiesa tai schema dirba testiniu režimu (sujungtos blogiausia su geriausia būsenos), o jei t yra netiesa tai schema dirba normaliu režimu.

4. Tiriamoji dalis

Šiame darbe tiriama programinės įrangos pateikiamų rezultatų tikslumas. Kadangi valdomumas yra skaičiuojamas remiantis atsitiktiniu duomenų generavimu, todėl gaunama paklaida. Šio darbo tikslas yra nustatyti kokia tai paklaida ir kaip ją būtų galima sumažinti. Taip pat tiriama kaip galima sutrumpinti skaičiavimų laiką, kad nenukentėtų rezultatų tikslumas jei vartotojas „neatsargiai“ pasirinks valdomumo skaičiavimo kriterijus.

4.1. Terminų žodynas

Pristatomi terminai naudojami tyrimo aprašyme:

- Iteracinis modelis – modelis sudarytas iš kombinacinių schemų ir imituojantis schemos su atmintimi darbą
- Iteracija – viena, iteracinį modelį sudaranti, automato kopija
- Pakartojimas – pakartotinis skaičiavimų atlikimas su iteraciniu modeliu
- Nuokrypis – rezultato procentinis skirtumas tarp iteracijų ar pakartojimų

4.2. Tyrimo prielaidos ir metodas

Valdomumo apskaičiavimo metodas remiasi atsitiktiniu duomenų generavimu. Dėl šios priežasties rezultatai yra pateikiami su tam tikra paklaida. Paklaida priklauso nuo iteracinio modelio kopijų skaičiaus, bei kiek kartų buvo įvykdytas iteracinis modelis – pakartojimų skaičiaus. Pradinėje sistemoje šiuos du parametrus turėjo įvertinti vartotojas, remdamasis automato įėjimų, išėjimų ir būsenų skaičiumi. Kadangi automato valdomumas priklauso ir nuo kitų charakteristikų, todėl vartotojui yra sunku ir praktiškai neįmanoma nustatyti minėtus parametrus kad būtų gaunamas tikslus rezultatas.

Šio tyrimo tikslas yra programiniu būdu nustatyti iteracijų bei pakartojimų skaičių, kad gauti tikslius rezultatus. Dėl šių parametrų nustatymo buvo pasirinkta naudoti panašų metodą – didinti jų skaičių kol rezultatas „nusistovi“ t. y. kai rezultatai nenukrypsta nuo ankščiau gautųjų.

Pakartojimų parametro prisisotinimas tikrinamas po kiekvieno iteracinio modelio įvykdymo. Žiūrima ar gautas bendras automato valdomumas nenukrypsta nuo prieš tai gautų penkiasdešimties rezultatų daugiau kaip dviem dešimtosiomis procento dalimis. Jei skiriasi daugiau nei 0,2% tada skaičiavimai tęsiami toliau. Jei per 50 paskutinių pakartojimų rezultatai nesvyravo virš šios ribos, tada galima teigti jog automato valdomumas, prie pasirinkto iteracijų skaičiaus, nusistovėjo ir netikslinga didinti pakartojimų skaičių, nes jie jau nebepakeis galutinio rezultato.

Iteracijų prisisotinimas nustatinėjamas kai jau yra nusistovėjęs pakartojimų skaičius. Pakeičiamas automatų iteracijų skaičius ir vėl skaičiuojamas jo bendras valdomumas. Gautas valdomumas yra lyginamas su prieš tai gautomis valdomumo reikšmėmis. Jos neturi svyruoti daugiau kaip dviem dešimtosiomis procento. Skaičiavimų pabaigos sąlyga ta pati kaip ir pakartojimų – jei per 50 paskutinių iteracijų pakeitimų valdomumas nesvyravo daugiau nei 0,2% tai reiškia kad iteracijų skaičiaus didinti nebereikia ir galima nutraukti skaičiavimus, nes rezultatai, gauti sekančių skaičiavimų metu, nebepakeis galutinio rezultato.

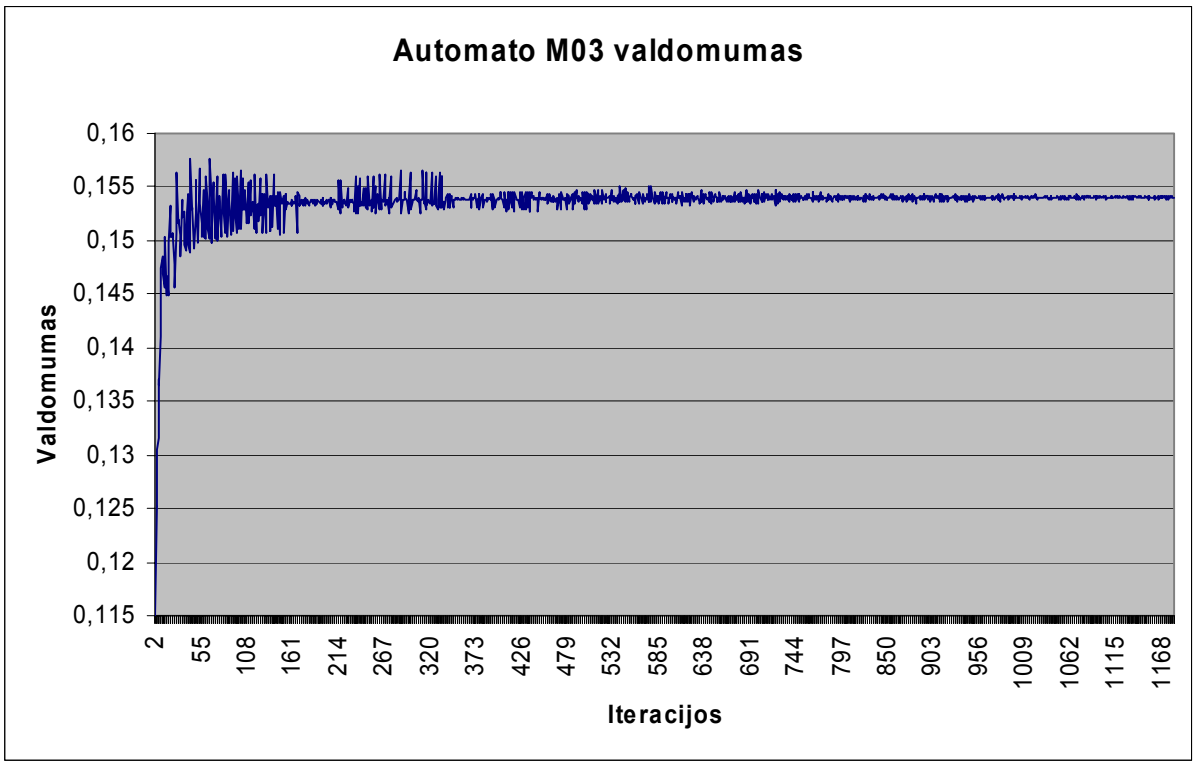
Tyrimo metu buvo iškeltos prielaidos:

1. Jei pakartojimų skaičius yra per mažas, tada gautas rezultatas bus netikslus, t. y. pakartotinai atlikus skaičiavimus su tuo pačiu automatu bus gautas kitoks rezultatas.
2. Jei iteracijų skaičius yra per mažas, tada gautas rezultatas taip pat gali būti netikslus, nes kai kuriems automatams kad pasiekti tam tikrą būseną reikia labai didelio iteracijų skaičiaus.
3. Pakartojimų skaičius turi priklausyti nuo iteracijų skaičiaus. Kuo didesnis iteracijų skaičius, tuo daugiau automato būsenų yra išnagrinėjama, tuo būdu yra gaunamas tikslesnis automato valdomumas. Kuo tikslesnis vieno pakartojimo vidurkis tuo mažesnis turi būti rezultatų svyravimas tarp pakartojimų.

Kad patvirtinti arba paneigti prielaidas buvo atlikti testai su abstrakčiais automatais. Buvo ištirti trys automatai. M03 ir M05 – automatai su atmintimi, ir C499 – automatas be atminties, todėl jo valdomumo pagerinimas nebus tiriamas.

4.3. Skaičiavimų tikslumo priklausomybė nuo iteracijų skaičiaus

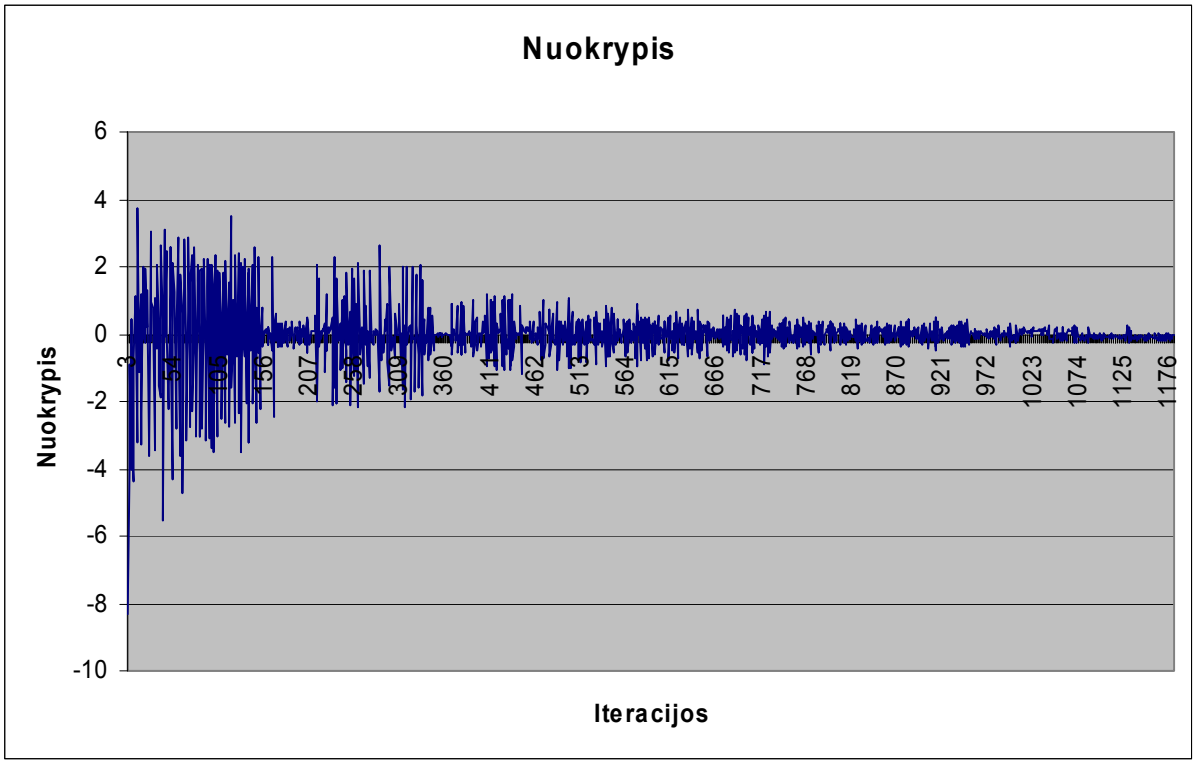
Automato M03 valdomumo priklausomybė nuo iteracijų skaičiaus pateikiama 17 pav.



17 pav. Automato M03 valdomumas

Skaičiavimai buvo nutraukti, kai buvo ištirtas iteracinis modelis su 1182 iteracijomis. Kadangi nusistovėjimo sąlyga tikrina nuokrypį 50 paskutinių rezultatų, todėl galima teigti kad automato M03 valdomumas nusistovi prie 1132 iteracijų.

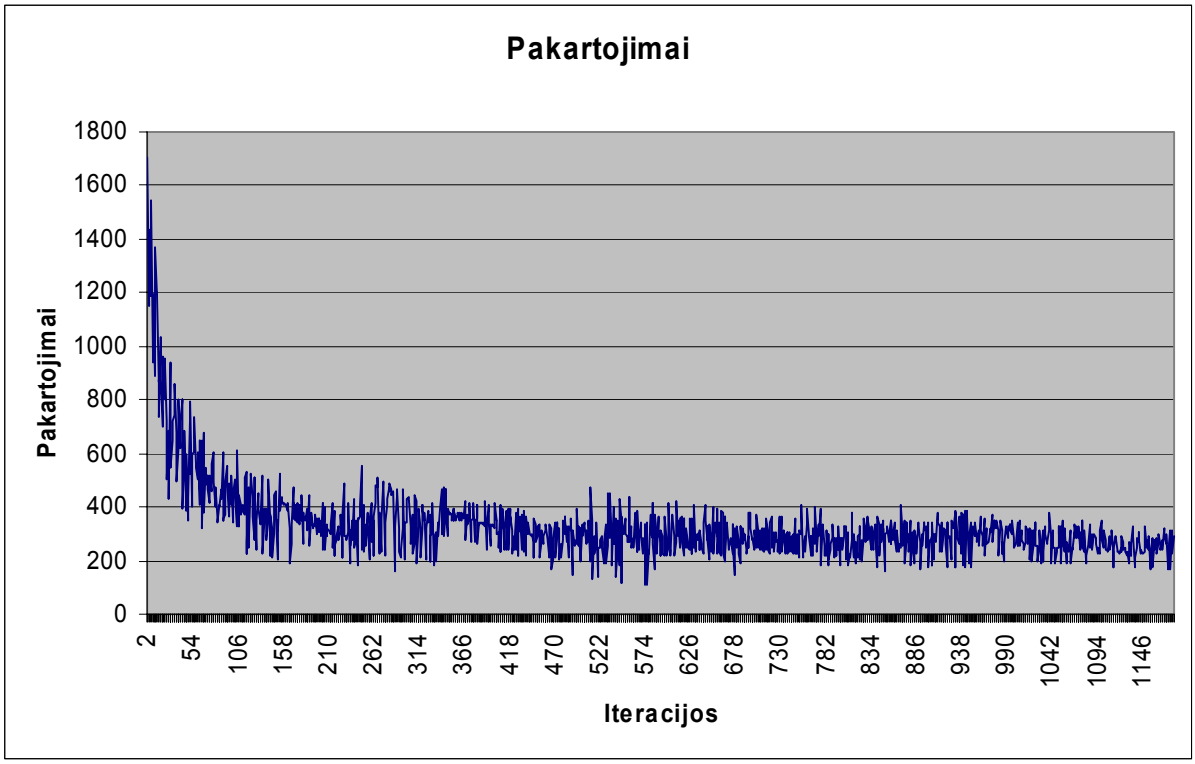
Nuokrypio priklausomybė nuo iteracijų skaičiaus yra pateikiama 18 pav.



18 pav. Automato M03 nuokrypis

Grafikas patvirtina kad skaičiavimai buvo nutraukti kai iteracijų rezultatai nesvyravo daugiau kaip 0,2%. Kad rezultatas tikslus patvirtina ir tai kad nuokrypis tolygiai mažėjo – tai nėra atsitiktinai gauti rezultatai.

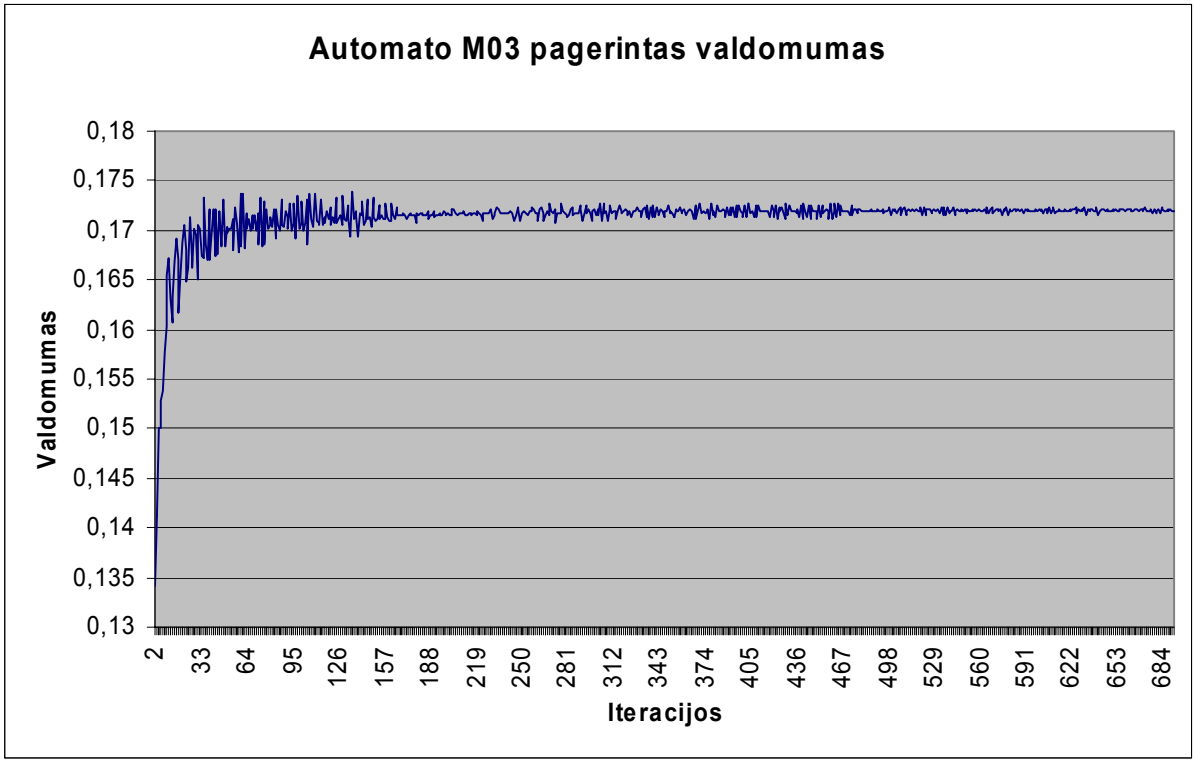
Pakartojimų skaičiaus priklausomybė nuo iteracijų skaičiaus pateikiama 19 pav.



19 pav. Automato M03 pakartojimai

Gauti rezultatai patvirtina trečią prielaidą, kuri teigia kad kuo didesnis iteracijų skaičius, tuo mažiau pakartojimų reikia. Šis efektas labiausiai pastebimas kai yra dar gana mažas iteracijų skaičius, o kuo toliau tuo mažiau mažėja pakartojimų skaičius. Pagal M03 automato pakartojimų skaičiaus priklausomybę nuo iteracijų skaičiaus galime teigti, kad pakartojimų skaičius eksponentiškai mažėja didėjant iteracijų skaičiui.

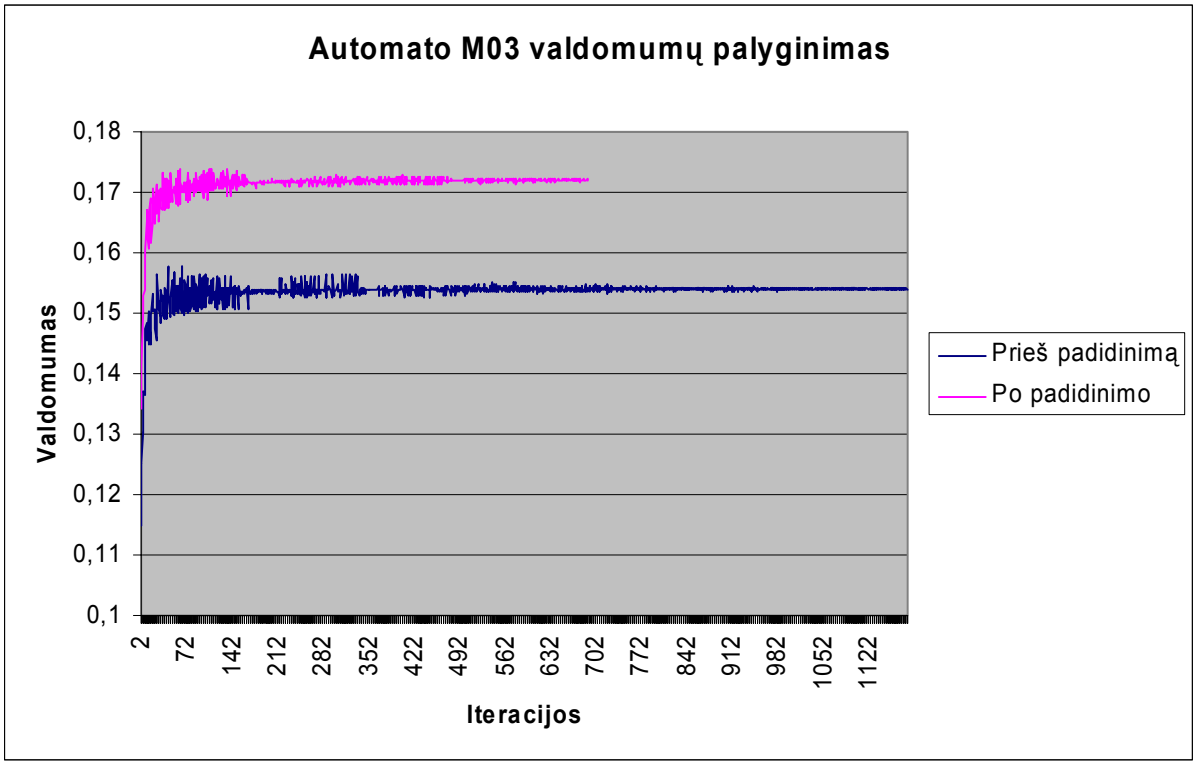
Tie patys eksperimentai buvo atlikti ir pagerinus M03 automato valdomumą. Pagerintas M03 automato valdomumas pateikiamas 20 pav.



20 pav. Automato M03 pagerintas valdomumas

M03 automato valdomumas kinta pagal panašią trajektoriją tiek prieš pagerinimą tiek ir po jo. Tačiau automatui su pagerintu valdomumu reikia dvigubai mažiau iteracijų. Bendras schemos valdomumas padidėjo 12%.

Valdomumo reikšmių palyginimas prieš padidinimą ir po pateikiamas 21 pav.

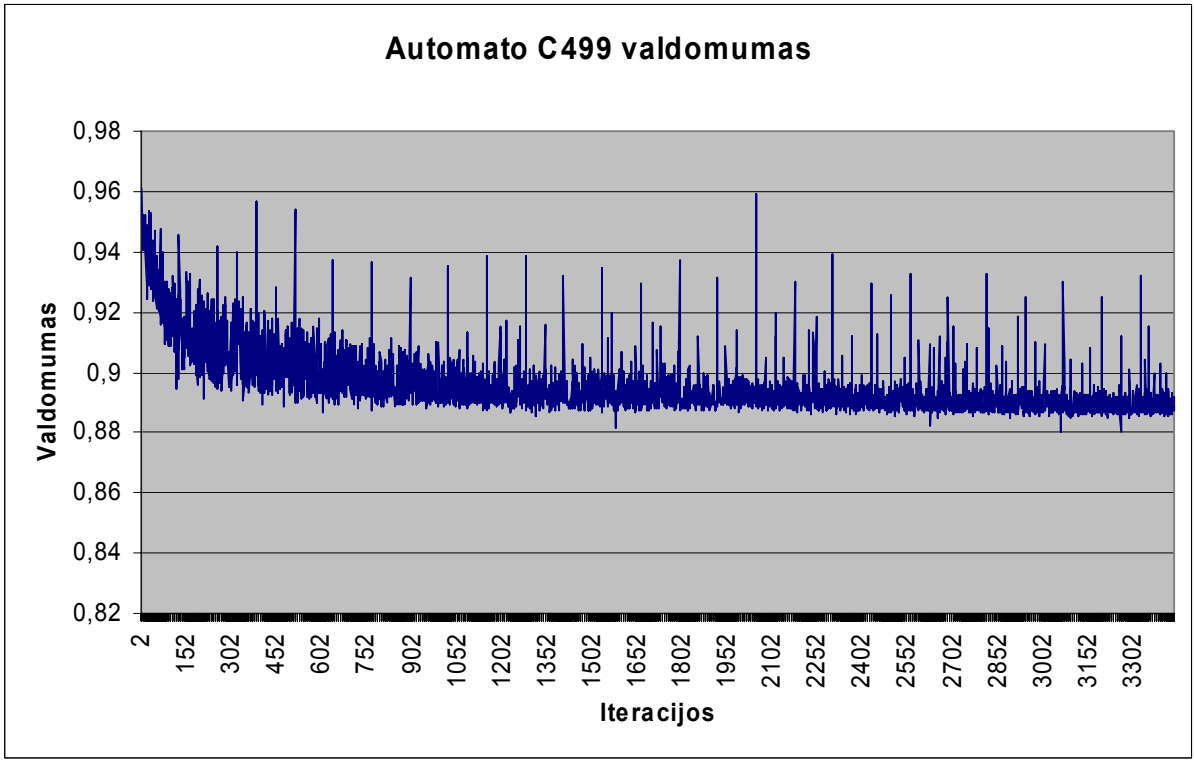


21 pav. Automato M03 valdomumų palyginimas

Automato, po valdomumo pagerinimo, M03 tyrimo rezultatai dėl nuokrypio ir pakartojimų skaičiaus minimaliai skiriasi negu kad prieš pagerinimą, todėl jie pateikiami 8.1 ir 8.2 prieduose.

Naudojant tuos pačius metodus buvo ištirtas M05 automatas. Jo tyrimo rezultatai pateikiami prieduose. Rezultatai iš esmės nesiskiria nuo M03 automato. Valdomumą pavyko pagerinti 6,49%.

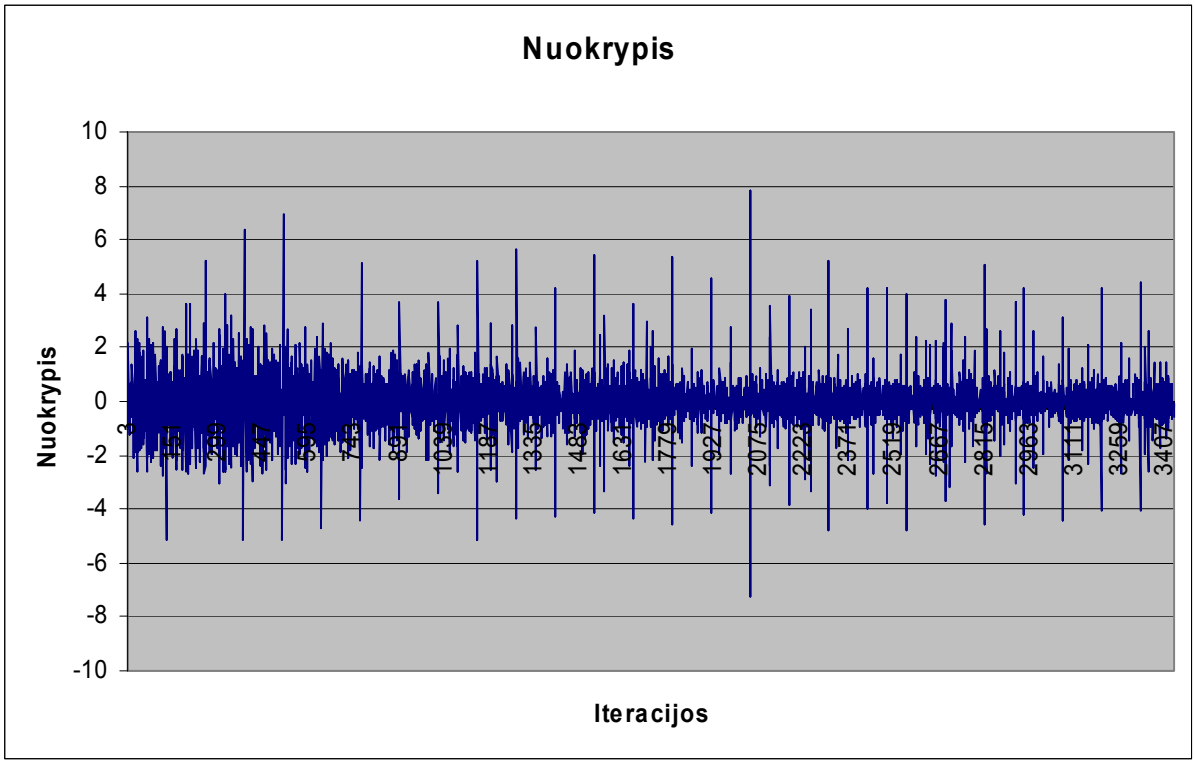
Taip pat buvo ištirta ir kombinacinė schema C499. Jos valdomumo grafikas pateiktas 22 pav.



22 pav. Automato C499 valdomumas

Šiam automatui reikėjo net 3389 iteracijų, kol jų skaičius nusistovėjo. Dar vienas skirtumas tarp kombinacinių ir schemų su atmintim – valdomumas kinta tolygiai kintant iteracijų skaičiui.

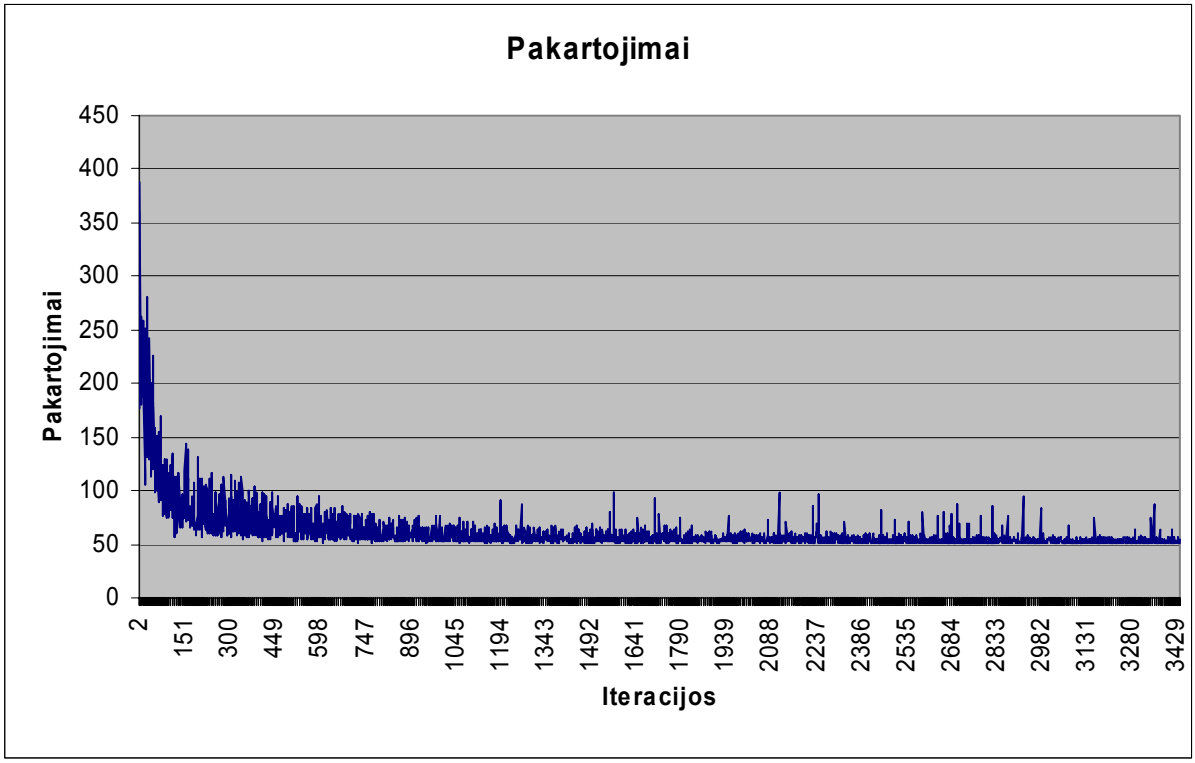
Automato C499 nuokrypio priklausomybė nuo iteracijų skaičiaus pateikta 23 pav.



23 pav. Automato C499 nuokrypis

Pagal gautus rezultatus galima teigti, kad skaičiavimų pabaigos sąlyga buvo atsitiktinai įvykdyta, t. y. valdomumo nusistovėjimas beveik nepriklauso nuo iteracijų skaičiaus kai nagrinėjama kombinacinė schema. Kombinacinėm schemom reikia koreguoti skaičiavimų pabaigos sąlygą.

Automato C499 pakartojimų skaičiaus priklausomybė nuo iteracijų skaičiaus pateikta 24 pav.



24 pav. Automato C499 pakartojimai

Pakartojimų skaičius, kaip tikėtasi, priklauso nuo iteracijų skaičiaus. Bet kadangi kombinacinė schema neturi vidinių būsenų, todėl nėra tikslo naudoti iteracinį modelį kombinacinėms schemoms. Nors naudojant iteracinį modelį, schemoms be atminties, negaunami klaidingi rezultatai, bet gali būti kad yra atliekami nereikalingi skaičiavimai.

Kadangi C499 schema yra be vidinių būsenų, o valdomumo pagerinimo metodas remiasi vidinių būsenų sujungimu, todėl valdomumo pagerinti negalima.

5. Išvados

1. Pristatyta testuojamumo problema, išanalizuoti įvairūs valdomumo nustatymo metodai, pristatyta taikymo sritis, apibrėžta automato valdomumo sąvoka, bei automatų imitavimo priežastys. Pristatyti valdomumo padidinimo metodai.
2. „Baltų dėžių“ valdomumo nustatymo ir padidinimo metodai adaptuoti „juodoms dėžėms“.
3. Sukurta PĮ įvertinanti automatų valdomumą, bei pateikianti pasiūlymus kaip galima padidinti valdomumą.
4. PĮ buvo dokumentuojama kuriant reikalavimų, architektūros, detalios architektūros specifikacijas ir vartotojo vadovą.
5. Tyrimo prielaidos, dėl pakartojimų ir iteracijų skaičiaus nustatymo, bei pakartojimų mažėjimo didėjant iteracijų skaičiui, buvo patvirtintos eksperimentų metu.
6. Valdomumo pagerinimo funkcija pagerina automato valdomumą apie 9,45%

6. Literatūros sąrašas

[1] K.Kuchcinski. Test Synthesis. Prieiga internete: http://www.cs.lth.se/home/Krzysztof_Kuchcinski/DES/Lectures/Lecture10.pdf [žiūrėta 2003 10 25].

[2] K.Cheng. Design for testability. Prieiga internete: http://cadlab.ece.ucsb.edu/~licwang/ece156b/dft.pdf. [žiūrėta 2003 11 03].

[3] M.Bushnell, V.Agrawal. Design for testability: Full scan. Prieiga internete: http://courses.cs.tamu.edu/cpsc680/walker/Slides/lec23.pdf. [žiūrėta 2003 11 03].

[4] T.Wheeler. Improving design observabiliy and controllability for functional verification of FPGA-based circuits using design-level scan techniques. Prieiga internete: http://splish.ee.byu.edu/docs/thesis-wheeler.pdf [žiūrėta 2003 10 21].

[5] Controllability, observability and testability. Prieiga internete: http://www.fe.up.pt/~allegro/wp3/d3x_wbm/MOD1/ControllabilityObservabilityTestability.html [žiūrėta 2003 10 21].

[6] Design for testability. Prieiga internete: http://www.pld.ttu.ee/diagnostika/labs/dftlab.htm [žiūrėta 2003 10 25]

[7] K.Kuchcinski. Test Synthesis. Prieiga internete: http://www.cs.lth.se/home/Krzysztof_Kuchcinski/DES/Lectures/Lecture10.pdf [žiūrėta 2003 10 25].

[8] Design for Testability. Prieiga internete: http://www.pld.ttu.ee/~raiub/web_0103/disain_ja_test/loengukiled/3_Testability_design.ppt [žiūrėta 2003 10 25]

[9] What is VHDL?. Prieiga internete: http://www.acc-eda.com/vhdlref/refguide/language_overview/language_overview.htm [žiūrėta 2003 10 25].

[10] K.Cheng. Design for testability. Prieiga internete: http://cadlab.ece.ucsb.edu/~licwang/ece156b/dft.pdf. [žiūrėta 2003 11 03].

[11] W.Huang, E.McCluskey. A Memory Coherence Technique for Online Transient Error Recovery of FPGA Configurations. Prieiga internete: http://crc.stanford.edu/crc_papers/huangfpga01.pdf. [žiūrėta 2003 11 03].

[12] M.Bushnell, V.Agrawal. Design for testability: Full scan. Prieiga internete: http://courses.cs.tamu.edu/cpsc680/walker/Slides/lec23.pdf. [žiūrėta 2003 11 03].

[13] Cheng-Wen Wu. Testability measures. Prieiga internete: http://www.te.tku.edu.tw/~chiang/courses/testing/Testing-2/tm.doc. [žiūrėta 2003 11 05].

[14] R.Daasch. Introduction to IC test. Prieiga internete: http://ece.pdx.edu/~daasch/course/x75/pdf/chapter6_2up.pdf. [žiūrėta 2003 12 08].

[15] Cheng-Wen Wu. Design for testability. Prieiga internete: http://larc.ee.nthu.edu.tw/~cww/n/625/6251/05DFT0110.pdf. [žiūrėta 2003 11 05].

[16] D.Kirovski, M.Potkonjak, L.Guerra. Imporving the observability and controllability of datapaths for emulation based debuging. Prieiga internete: http://citeseer.nj.nec.com/cache/papers/cs/23539/http:zSzzSzwww.cs.ucla.eduzSz~darkozSzpaperszSzdebugASIC.pdf/kirovski99improving.pdf [žiūrėta 2003 10 15].

[17] T.Wheeler, P.Graham, B.Nelson, B.Hutchings. Using Design-Level Scan to Improve FPGA Design Observability and Controllability for Functional Verification. Prieiga internete: http://citeseer.nj.nec.com/cache/papers/cs/23239/http:zSzzSzsplish.ee.byu.eduzSzdocszSzfpl01.scan.pdf/wheeler01using.pdf [žiūrėta 2003 12 08].

[18] P.Graham, B.Nelson B. Hutchings. Instrumenting Bitstreams for Debugging FPGA Circuits. Prieiga internete: http://citeseer.nj.nec.com/cache/papers/cs/23239/http:zSzzSzsplish.ee.byu.eduzSzdocszSzfccm01.bitstream-instr.pdf/graham01instrumenting.pdf [žiūrėta 2004 11 05].

[19] P.Folkesson, S.Svensson, J.Karlsson. A Comparison of Simulation Based and Scan Chain Implemented Fault Injection. Prieiga internete: http://www.ce.chalmers.se/~johan/publications/ftcs28camera.frm.pdf [žiūrėta 2004 11 07]

7. Santrumpų ir terminų žodynas

Valdomumas – matas nusakantis kaip lengva ar sunku nustatyti pasirinktą schemos mazgą į 0 arba 1

Iteracinis modelis – modelis sudarytas iš kombinacinių schemų ir imituojantis schemos su atmintimi darbą

Iteracija – viena, iteracinį modelį sudaranti, automato kopija

Pakartojimas – pakartotinis skaičiavimų atlikimas su iteraciniu modeliu

CGI – (*Common Gateway Interface*) kliento serverio sąsaja

PHP – programavimo kalba skirta dinaminiam interneto puslapių kūrimui

HTML – (*Hypertext markup language*) interneto puslapių kūrimo kalba

FTP – (*File Transfer Protocol*) failų perdavimo protokolas

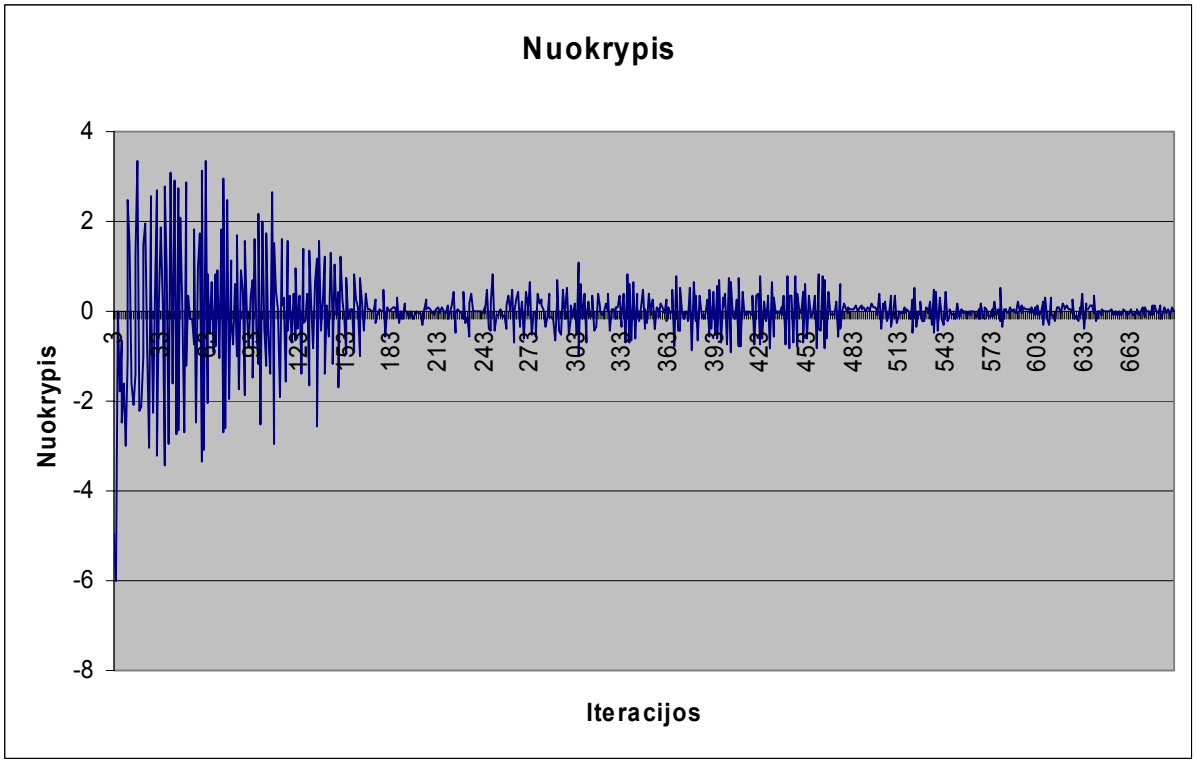
PDF – (*Portable Document Format*) dokumento formatas skirtas įvairioms sistemoms

gcc – UNIX sistemos C\C++ kalbos kompiliatorius

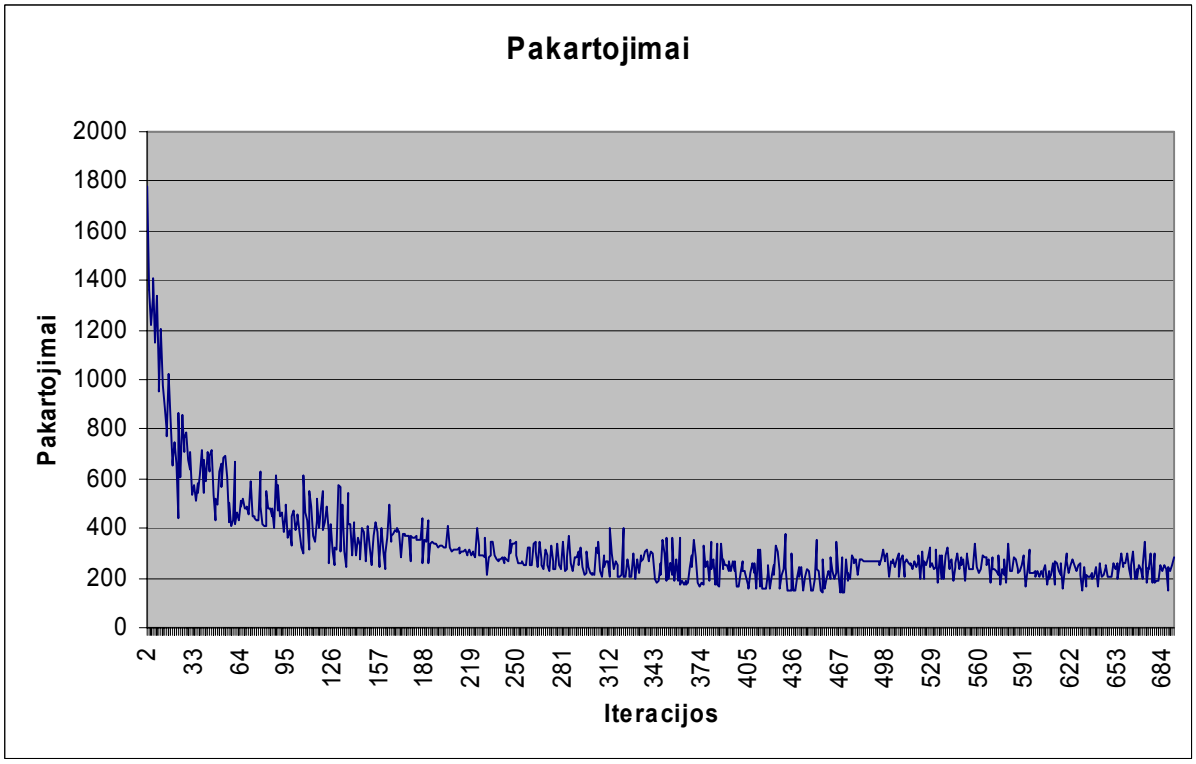
Serveris (darbo stotis) – pagrindinis tinklo kompiuteris (tarnybinė stotis), kuriame patalpinta programinė įranga.

8. Priedai

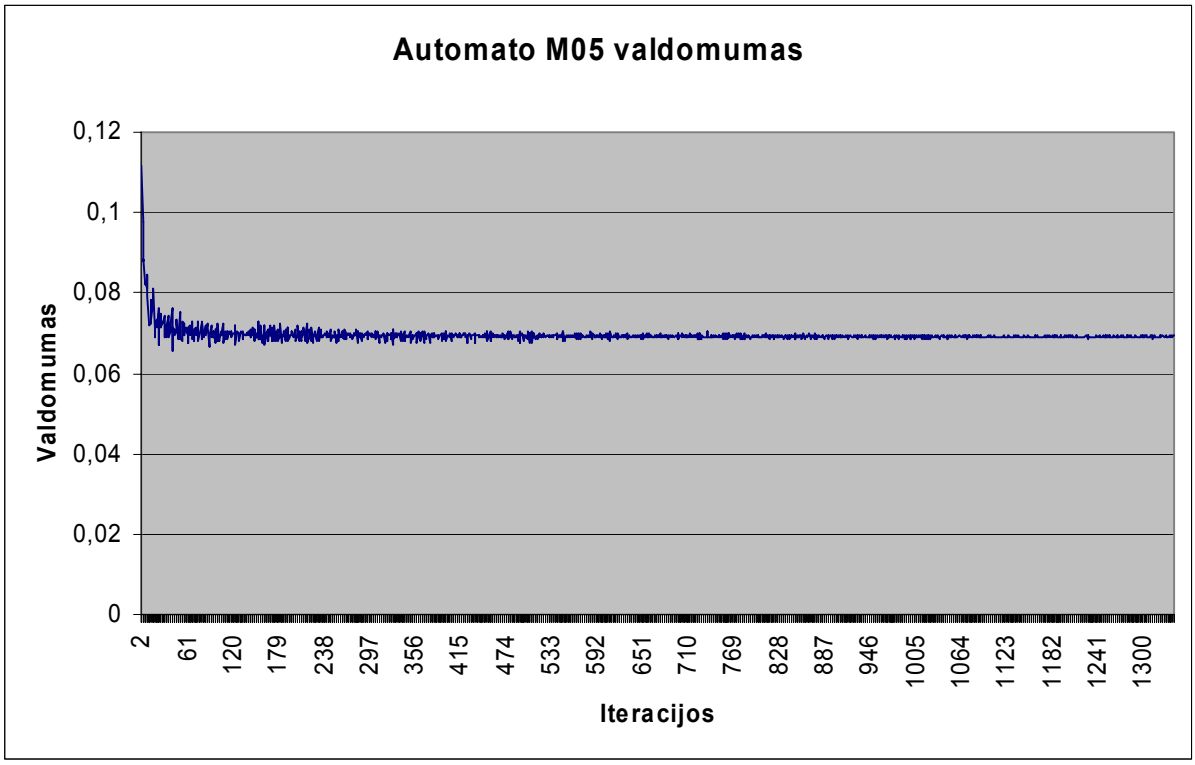
8.1. Automato M03, su pagerintu valdomumu, nuokrypis



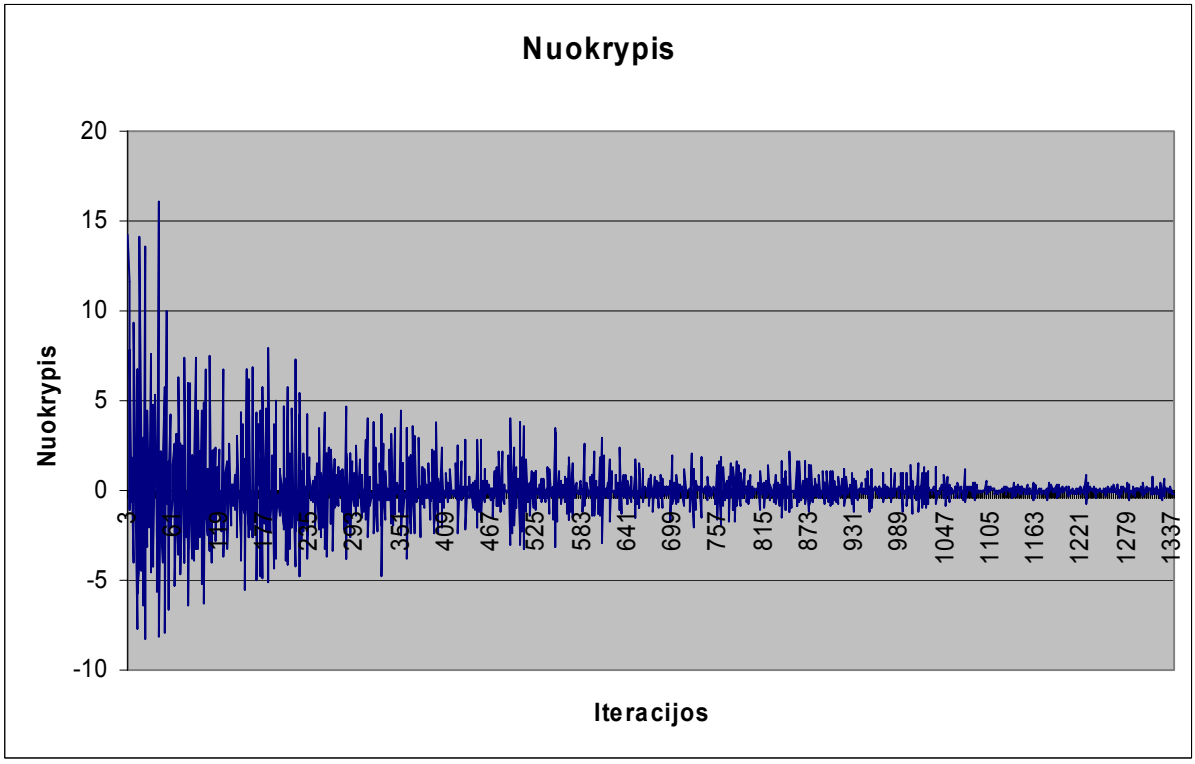
8.2. Automato M03, su pagerintu valdomumu, pakartojimai



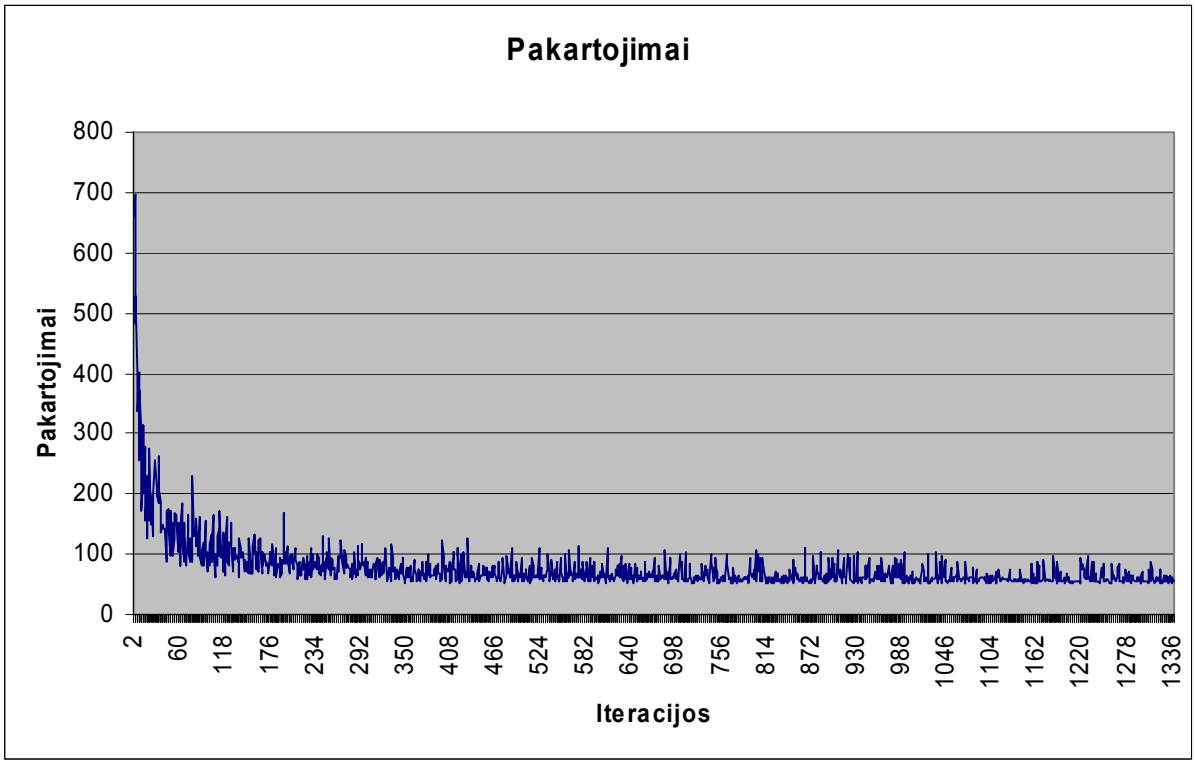
8.3. Automato M05 valdomumas



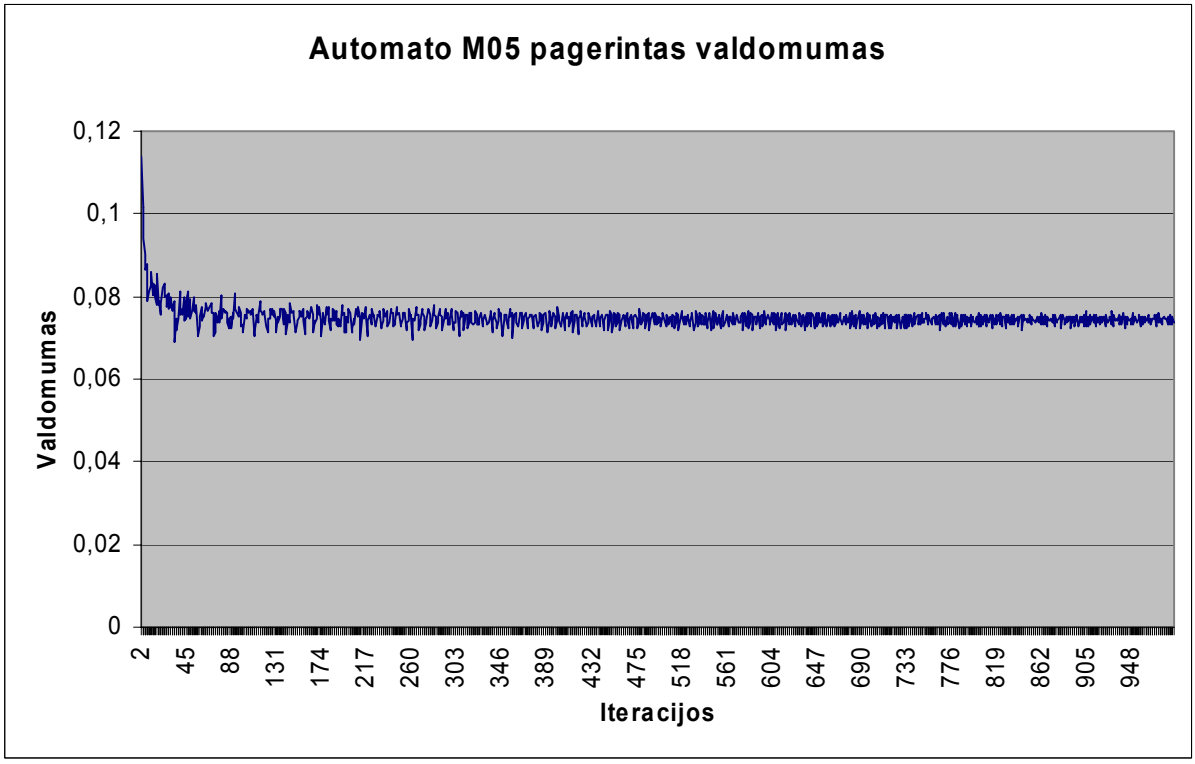
8.4. Automato M05 nuokrypis



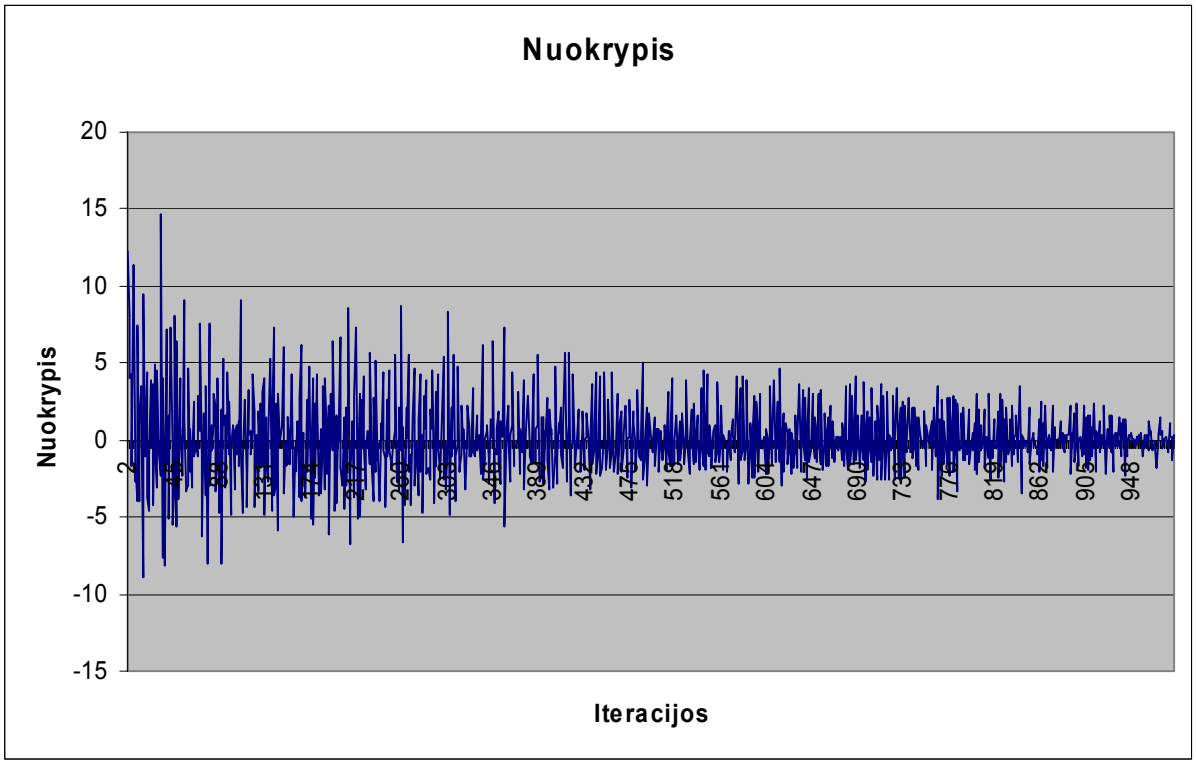
8.5. Automato M05 pakartojimai



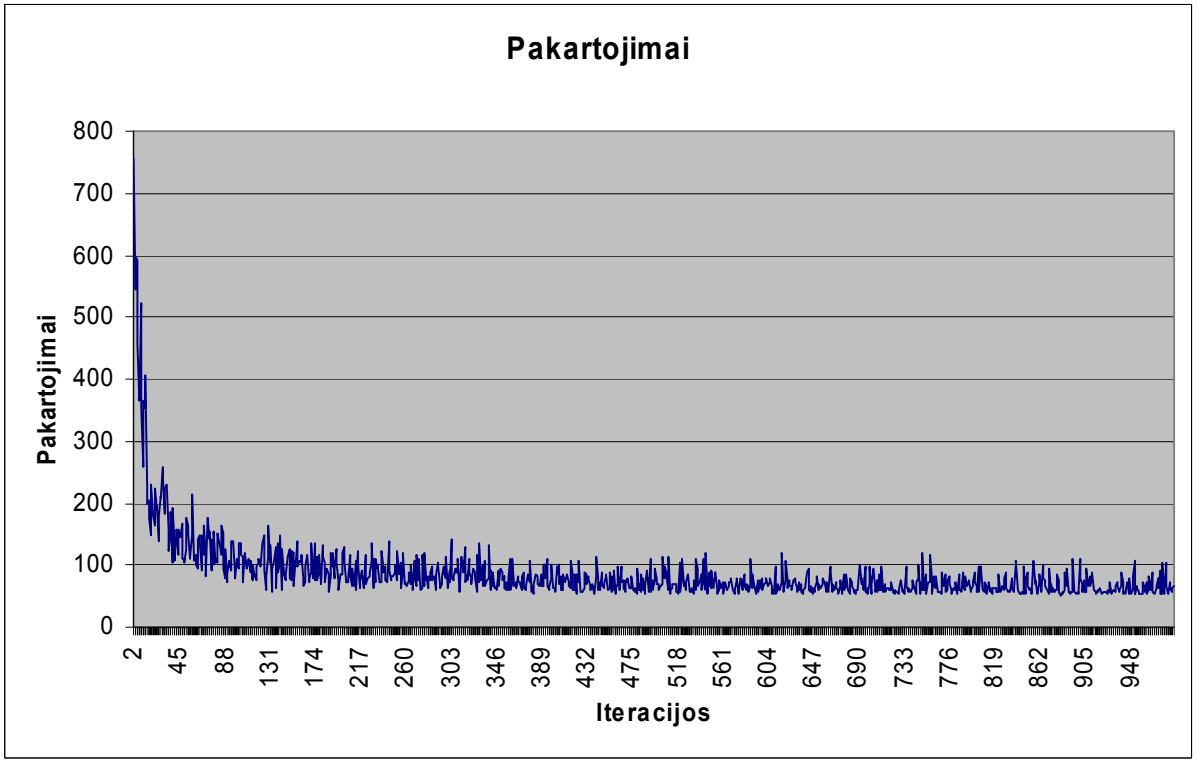
8.6. Automato M05 pagerintas valdomumas



8.7. Automato M05, su pagerintu valdomumu, nuokrypis



8.8. Automato M05, su pagerintu valdomumu, pakartojimai



8.9. Automato M05 valdomumo palyginimas