KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
PROGRAMŲ INŽINERIJOS KATEDRA

Martynas Šimkevičius

Funkcinių testų metodų vėlinimo gedimams tikrinti sudarymas ir tyrimas

Magistro darbas

Recenzentas
doc. A.Lenkevičius
2007-05-

Vadovas
prof. V.Jusas
2007-05-

Atliko
IFM-1/5 gr. stud.
Martynas Šimkevičius
2007-05-

Kaunas, 2007

Šimkevičius M. Funkcinių testų metodų vėlinimo gedimams tikrinti sudarymas ir tyrimas: Programų inžinerijos magistro darbas / vadovas prof. V.Jusas; Kauno technologijos universitetas, Informatikos fakultetas, Programų inžinerijos katedra. – Kaunas, 2007.- 53 p.

SANTRAUKA

Magistro darbe analizuojamas funkcinio testo sudarymas aukštame abstrakcijos lygmenyje. Toks testo kūrimo būdas leidžia vystyti testą ankstyvuose projektavimo etapuose lygiagrečiai su kitomis projektavimo proceso veiklomis. Realizuojamas funkcinis vėlinimo testo sudarymo algoritmas pagal pokyčius išėjimuose, kuris remiasi tik programinio prototipo pirminių įėjimų ir pirminių išėjimų reikšmėmis. Atliekami eksperimentai. Eksperimento metu atliekami testinių rinkinių generavimai 35 loginėms schemoms pasinaudojant 15 skirtingos spartos procesorių.

Gauti testo greitaveikos bei kokybės rezultatai parodė jog algoritmas pakankamai spartus ir gebantis rasti visus reikalingus rinkinius funkcinio vėlinimo testo sudarymui.

Šimkevičius M. Creation and investigation of functional delay test : Master's Work in Software Engineering / supervisor prof. V.Jusas; Department of Software Engineering Faculty of Informatics Kaunas University of Technology. – Kaunas, 2007. – 53 p.

SUMMARY

This work studies the test generation in functional level of the circuits. Such an approach allows developing the test at the early stages of the design process in parallel with other activities of this process. Created test generation algorithm of the functional test that is based solely on the primary input values and the primary output values of the programming prototype. The experiment contains 35 logic schemas processed test generation algorithm on 15 different powers CPU.

The results of experiment shows that suggested algorithm is capable to find all needed pairs to complete functional test and the speed of algorithm is measurable at the same time.

Santrumpų ir terminų žodynas

ALU – aritmetinis loginis įtaisas (*angl. Arithmetic Logic Unit*)

SoC – vienlustės sistemos (*angl. System on Chip*)

AND – loginė daugyba

OR – loginė sudėtis

NAND – paneigta loginė daugyba

NOR – paneigta loginė sudėtis

XOR –

PII 350 – Pentium II (Deschutes) 350 MHz

PII 600 – Pentium III (Katmai) 600MHz

C800 – Celeron (Coppermine) 800MHz

D800 – AMD Duron(tm) processor 800MHz

C1300 – Intel(R) Celeron(TM) CPU 1300MHz

C1700 – Intel(R) Celeron(R) CPU 1.70GHz

S2500 – AMD Sempron(tm) 2500+

XP2000 – AMD Athlon(tm) XP 2000+

CD1666 – Genuine Intel(R) CPU T2300 @ 1.66GHz (1 branduolys)

X5110 – Intel(R) Xeon(R) CPU 5110 @ 1.60GHz (1 branduolys)

C2800 – Intel(R) Celeron(R) CPU 2.80GHz

P4D3000 – Intel(R) Pentium(R) D CPU 3.00GHz (1 branduolys)

C2D1866 – Intel(R) Core(TM)2 CPU 6300 @ 1.86GHz (1 branduolys)

A64X2 3800 – AMD Athlon(tm) 64 Processor 3800+

C2D3000 – Intel(R) Core(TM)2 CPU 6300 @ 3.0 GHz (1 branduolys)

Lentelių sąrašas

Paveikslų sąrašas

1. Įvadas

Darbo aktualumas

Visos naujos integrinės schemos prieš patekdamos į gamybą yra testuojamos funkciniame lygmenyje. Tam naudojami funkciniai testai kurie remiasi schemoje vykdoma funkcija, kuri gali būti realizuota daugeliu būdu. Schemos galimi gedimai priklauso nuo schemos realizacijos. Praktikoje apsiribojama schemos konkrečios realizacijos galimais gedimais ir testas sudaromas tik vienos realizacijos gedimams. Testą galima sudaryti tik turint konkrečią schemos realizaciją. Tuo tarpu funkcinis testas nepririštas prie konkrečios realizacijos. Tokiu atveju funkcinis testas turi tikrinti visas galimas realizacijas,o tai yra žymiai sudėtingesnė problema. Pagrindinė problema su kuria susiduriama sudarant funkcinį testą kaip vertinti funkcinio testo kokybę neturint konkrečios schemos realizacijos. Šiuo metu jau yra pasiūlyti funkcinių gedimų modeliai, kurie leidžia gauti funkcinį testą, kuris tikrina daugiau kaip 99% bet kokios realizacijos konstantinių gedimų. Bet toks funkcinis testas yra keletą kartų ilgesnis negu testas paskaičiuotas konkrečiai schemos realizacijai. Todėl funkcinis testas dar turi būti minimizuotas konkrečios realizacijos gedimų atžvilgiu išmetant iš jo testinius rinkinius, kurie netikrina naujų konkrečios realizacijos gedimų. Be to neatmetama galimybė esant aukštiems testo pilnumo reikalavimams funkcinį testą papildyti netikrinamų konkrečios realizacijos gedimų atžvilgiu. Funkcinio testo naudojimo pagrindinis privalumas yra tas, kad funkcinį testą galima projektuoti pradiniuose schemos projektavimo etapuose pagal schemos programinį prototipą lygiagrečiai su schemos sintezės procesu. Tuo tarpu funkcinio testo minimizavimas besiremiantis jau po sintezės gauta konkrečia realizacija nėra darbui imlus veiksmas ir mažai įtakoja bendrą schemos sintezės ir testų projektavimo trukmę.

Darbo tikslas – apžvelgti funkcinių testų, vėlinimo gedimams nustatyti, sudarymą funkciniame lygmenyje ir realizuoti algoritmą gebanti atrinkti testinius rinkinius, pagal pokyčius išėjimuose, funkcinio vėlinimo testo sudarymui.

Uždaviniai:

1) Išanalizuoti algoritmus skirtus funkcinių vėlinimo testų sudarymui;
2) Realizuoti algoritmą gebanti atrinkti testinius rinkinius funkciniam testui sudaryti pagal pokyčius schemos išėjimuose;
3) Ištirti algoritmo gebėjimą rasti reikiamus rinkinius;
4) Ištirti algoritmo greitaveiką analizuojant pasirinktas schemas;

Darbo metodika

Darbą sudaro įžanga, trys skyriai: *2 Probleminės srities apžvalga*, *3 Metodas 4 Tyrimo rezultatai ir jų įvertinimas*; išvados, naudotos literatūros sąrašas ir priedai.

Darbo apimtis 77 puslapiai, jame yra 5 lentelės, 104 paveikslai, literatūros sąrašą sudaro 26 šaltiniai.

Pirmame etape trumpai apžvelgiama egzistuojantys vėlinimo testų sudarymo algoritmai.

Antrame etape, apžvelgiama realizuoto algoritmo struktūra.

Trečiame etape pateikiami ir įvertinami gauti rezultatai.

Turint rezultatus ir analizę, ketvirtame etape skelbiamos išvados.

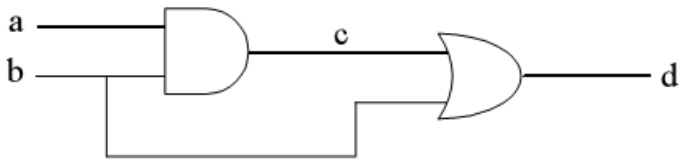
Algoritmas realizuojamas C++ kalboje. Algoritmo testavimas atliekamas naudojant Linux OS, kodas kompiliuojamas naudojantis gcc 3.xx įrankiais.

Probleminės srities apžvalga

2.1. Klasikiniai metodai

Ši ilgiausio kelio vėlinimo suradimo problema jau yra seniai nagrinėjama ir analizuojama [4][5][6][7][8]. Nors kažkurie uždelsimo defektai yra išsibarstę pačiame kelyje, kaip tranzistoriaus kanalo nukrypimas pagamintoje schemoje. Klaidos, sukeltos šių padarinių yra vadinamos visuotinėmis vėlinimo klaidomis [11]. Laiko ir jėgos optimizavimas linkę sumažinti kelių uždelsimą schemose, todėl daugelis kelių yra artimi maksimaliam uždelsimui [12]. Juo labiau, kad proceso variacijos atsiranda bet kurioje integrinės schemos vietoje, netgi pavieniame ventilyje, dėl to yra labai sudėtinga nustatyti kuri iš visų kelių yra iš tikrųjų ilgiausia. Taipogi testuojant tik vieną kelią negalima garantuoti, kad bus aptikta mažiausia uždelsimo klaida. Testuojant ilgiausias K kelias didėja klaidos aptikimo tikimybė, kadangi tai padidina tikimybę, jog tikrinama ilgiausias kelias. Yra teigiama, kad kelias gali būti testuojamas jeigu kylantis/krentantis frontas gali perduoti duomenis iš pirminės įvesties į pirminę informacijos išvestį, visa tai surišta su keliu [1][2][13][14][15]. Jeigu kelias yra netestuojamas, jis yra vadinamas netestuojamu arba klaidingu keliu [16].

Pvz., 1 paveiksle kelias a-c-d yra klaidingas remiantis vienpusio kelio jautrumo kriterijais [13], kadangi norint perduoti duomenis per AND ventilį reikia, kad linija b būtų logiškai lygi 1, analogiškai norint perduoti duomenis per OR ventilį reikia jog linija b būtų logiški lygi 0.



1 pav. Klaidingas kelias

Ankstesni tyrimai [4][5][6][7] apie ilgiausią kelią, einantys per kiekvieną įėjimą, yra neefektyvūs arba paprasčiausiai nesugeba garantuoti generuojamų kelių testavimo. Nepakankamas naudingumas atsiranda dėl to, jog iš pradžių ieškomi ilgiausi keliai ir tik po to autoriai kalba apie jų testavimą. Jeigu poschemėse, kurias kerta daug kelių, visi ilgiausieji keliai yra panašus, tai yra didelė tikimybė, jog nei vienas iš jų nėra testuojami.

2.2. NEST algoritmas

Daugelis ATPG buvo jau analizuojami, norint nustatyti ar jie galėtų būti patobulinti, norint rasti K ilgiausią testuojamą kelią per kiekvieną grandinės ventilį (KLPG). Ganėtinai greitas ATGP įrankis NEST [17] generuoja kelius neapskaičiuojamu būdu, kuris gali vienu laiku apdoroti didelį kelių kiekį, tačiau jis efektyvus tik tuo atveju jeigu yra testuojamos schemos, kuriuose yra daug testuojamų kelių su uždelsimo klaidomis.

2.3. DYNAMITE ir RESIST algoritmai

DYNAMITE [18] yra labai veiksmingas su sunkiai testuojamomis schemomis, tačiau lengvai testuojamose schemose klaidos yra traktuojamos atskirai, o tai sukelia didelį atminties suvartojimą ir nėra labai praktiška dideliems schemoms. Pagal RESIST [19] algoritmą daugelis kelių schemose turi daug bendrų sub kelių ir naudojais šias sub kelias tik kartą, o tai sumažina pasikartojantį darbą ir identifikuoja didelius netestuojamų kelių kiekius. Maža to per pirmą analizės kartą šis būdas identifikavo apie 99.4 % visų vėlinimo kelių, ir testuojamoje ir netestuojamoje schemoje c6288, kuri yra žinoma turinti proporcingai didėjantį kelių skaičių. Nepaisant visko, testavimo greitis c6288 yra vis ganėtinai mažas. RESIST reikėjo vienos 122 procesorių valandos norint aptikti 12592 kelių SPARC IPX 28 MIPS.

2.4. Patobulintas RESIST

Paskutiniai tyrimai [8] pristatė efektyvų problemos sprendimo metodą. Šis metodas išplečia RESIST algoritmą ieškant ilgiausią testuojamą kelią per kiekvieną ventilį. Šis metodas gauna naudos iš ryšių tarp ilgiausių kelių per skirtingus ventilius ir garantuoją jų testavimo galimybes. Iš dalies dėl to jog šis darbas naudoja elementų vėlinimo modelį, kuriame nėra akivaizdžių būdų galinčių išplėsti metodą iki to kad išspręsti problemą ieškant ilgiausio testuojamo kelio K einančio per kiekvieną ventilį, be to šis metodas nepasiteisina testuojant c6288. Įvertinant šiuos trukumus esančius RESIST algoritme, matome jog RESIST algoritmas ne visada gali būti lengvai pritaikomas generuojant kelius kiekvieną ventilį.

2.5. Numanomo neteisingo kelio eliminavimo algoritmas

Vėlinimo analizės priemonė paremta nežinomų intervalų tyrimų [3] pristato kitokį metodą, kuris gali efektyviai identifikuoti ilgiausią testuojamą kelią kombinuotuose keliuose. Vietoj to, kad generuotų struktūrinis ilguosius kelius ir tikrintų jų testavimą galimybes, šis metodas analizuoja kelius nuo pat jų pirminių įvesties vietų. Kiekvienoje iteracijoje prie konstrukcijos pridedamas naujas ventilis ir jam pritaikomi įvairūs apribojimai. Vietoje to jog priskirti logines reikšmes vienam ar keliems pirminiams įėjimams, kad patenkinti apribojimus naujai pridėtiems ventiliams, kaip tai padaryta VIPER metodu [15], tiesioginė implikacija, kuri labiau efektyvi, yra pritaikoma lokalių konfliktų paieškai. Jei konfliktas egzistuoja, visa paieškos zona, kuri turi jau sudėtas ventilių grandines, yra pašalinama lauk. Šis metodas vadinamas numanomo neteisingo kelio eliminavimu [13][20]. Kai kurios kitos negalimų kelių eliminavimo technologijos kaip išankstinis šalinimas ar dinaminiai šalintojai yra taip pat pritaikomos negalimų kelių nustatymui prieš tikrinimą. Šis įrankis yra pakankamai efektyvus, jog jo pagalba butu galima testuoti c6288.

2.6. Globalaus ilgiausio kelio generavimas

Globalus ilgiausias kelias yra ilgiausio kelias per visą schemą, nepaisant to, kuriuos ventilius jis kerta. Globalaus ilgiausio kelio algoritmas yra kelio generavimo algoritmo tam tikram ventiliui modifikacija. Jei nėra ventilių kurie pašalinami iš dalinio kelio, tai pilnas kelias sudarytas iš dalinių kelių ir yra globaliai ilgiausias kelias.

Šio modelio privalumas: jeigu p yra globaliai ilgiausi keliai dengintys ventilius g_i, tai šie keliai turi būti p ilgiausi keliai einantys per g_i. Nors, pradžioje, globaliai ilgiausio kelio generavimas gali pereiti daugumą ventilių. Palyginimui, jeigu nėra atliktas globaliai ilgiausio kelio generavimas, tai ventiliai g_i taip pat įgyja ilgiausio kelio einančio per jį potencialą, ilgiausio kelio generavimo procese kitiems ventiliams, bet daugumoje atvejų jie ne bus testuojami, kol kelio generavimas skirtas g_i nebus atliktas

Kita vertus kuo globaliai ilgiausio kelio generavimas randa daugiau kelių, tuo galimybė jog kelias dengia dauguma ventilių mažėja. Blogiausiu atveju dauguma globaliai ilgų kelių praeina tik per labai mažas ventilių grupeles. Tai pat pradedant globalaus ilgiausio kelio generavimą yra naudinga, o po kelių rastų kelių netgi būtina nustatyti kelių generavimą per konkrečius ventilius.

Čia naudojama dviejų fazių strategija. Pirmiausia ieškoma globaliai ilgiausių kelių kol neliks nė vieno svarbaus ventilio. Po to generuojami individualūs keliai ventiliams kurie nėra pilnai padengti keliais iš globalių kelių generavimo.

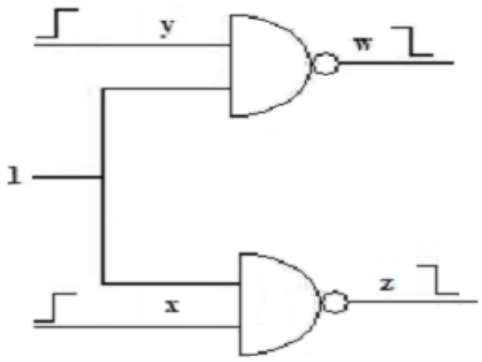
Viso to privalumas yra tame, jog globaliai ilgiausio kelio generavimas ne tik atmeta kažkuriuos ventilius nuo individualaus kelio generavimo, tai taip pat smarkiai pagreitina individualių kelių generavimą tiems ventiliams kuria neatmetami. Individualių ventilių kelių generavime visi daliniai keliai, su mažiausiais naudingais keliais, yra ilgesnis už ilgiausia globalų kelią gali būti pašalinamas, kadangi visi testuojami keliai kuriu ilgis didesnis nei ilgiausias globalus kelias yra jau sugeneruoti. Ši technologija ir ypatingai naudinga kai testuojama schema turi labai daug ilgų klaidingų kelių.

2.7. Perdavimo vėlinimo gedimų testai

Perdavimo vėlinimo gedimų testai yra dalinai panašūs į blogų ventilių paieškos testus. Blogų ventilių testai pritaikyti ieškoti defektų jau pagamintuose lustuose. Jie aptinka schemos klaidas sukeltas nutrukus jungimui, susijungimui su maitinimu ar žeme ir vidinių ventilių broku, kai ventilis nesugeba keisti savo reikšmės, o vietoje to pastoviai išduoda logini vienetą arba nulį. Perdavimo vėlinimo gedimai pasireiškia bet kuriame schemos taške, kuriame yra krentantis arba kylantis frontas, tai gali būti trigerio išėjimas ar net visos schemos, ar jos dalies išėjimas. Pagal prigimtį šie gedimai gali būti lėto pakilimo ar lėto nukritimo klaidos. Lėto pakilimo klaidos pasireiškia tada kai schemai dirbant maksimaliu dažniu, reikiamu momentu, loginis nulis nespėja pavirsti loginiu vienetu. Analogiškai, lėto nukritimo klaidos pasireiškia kai loginis vienetas nespėja virsti loginiu nuliu.

Betkurioje schemoje kelio vėlinimas gali būti nusakomas kaip skirtumu tarp generavimo pradžios kai signalas pradedamas skleisti, ir to momento, kai signalas suaktyvina signalo gavėją. Ventilių lygmens vėlinimo klaidos pasireiškia ventilio išėjime, o vėlinimas turi būti toks, kad bent vienas ventilio išėjimas turi nespėti sureaguoti į įėjimo pokytį. O algoritmas, ieškantis visų kelių einančių per vėlinimo klaidas sukeliančią schemos dalį, vadinamas plataus vėlinimo algoritmu [22].

Visos testo dalys, kurios sėkmingai nustato perdavimo vėlinimo klaidas, susideda iš dviejų vektorių {V1, V2}, kur V1 yra priminis vektorius, nusakantis pradines reikšmes pasirinktiems taškams ir V2 - įvykio vektorius, kuris ne tik aktyvuoja perėjimus tiriamuose elementuose, bet ir iššaukia pokyčio efektą viename ar keliuose išėjimuose[23]. Pavyzdžiui paimkime mažą grandinę pavaizduota 2 paveiksle:



2 pav. Dviejų ventilių grandinė

Grandinė turi du neigiamos daugybos ventilius kurie turi po vieną bendrą ir atskirą įėjimus. Panaudojus du vektorius įėjimuose x ir y sukuriame kylanti frontą. Pokytis įėjimuose sukelia pokyti išėjimuose w ir z, taigi mes stebime 4 galimus vėlinimo gedimus: x,y,w,z kojoms.

2.8. Funkcinio vėlinimo testo kokybės vertinimo kriterijai

Vėlinimo gedimai tikrinami rinkinių pora. Rinkinių poros, signalų reikšmės, apsprendžia, kurie įėjimai keičiasi ir kurie išlieka pastovūs. Gali keistis vienas ar daugiau įėjimų. Pokyčiai įėjimuose, kurie iššaukia pokyčius ir išėjimuose, gali patikrinti vėlinimo gedimus. Pokyčiai iš schemos įėjimų į išėjimus perduodami schemos keliais. Vėlinimo gedimų tikrinimui tikslinga pokyčius iš įėjimų į išėjimus perduoti kuo ilgesniais ir kuo įvairesniais keliais. Jeigu pokytis įėjime įtakoja pokytį išėjime, tai galima daryti prielaidą, kad gali būti patikrinti kai kurie vėlinimo gedimai. Funkcinio testo kokybės vertinimui svarbu nustatyti, koks įėjimo pokytis kokius išėjimus įtakoja. Kai keičiasi daug įėjimų tai nėra paprastas uždavinys. Todėl tikslinga nagrinėti atskirų įėjimo pokyčių panaikinimo įtaką išėjimo pokyčiams. Kalbėsime apie atskiro įėjimo pokyčio panaikinimą, kai šio įėjimo antro rinkinio reikšmė prilyginama pirmo rinkinio reikšmei.

Laikykime, kad turime rinkinių porą {1010,0111}, kur keičiasi pirmo, antro ir ketvirto įėjimų reikšmės. Panaikinant pokytį pirmam įėjime, gauname rinkinių porą {1010,1111}, panaikinant pokytį antram įėjime, gauname rinkinių porą {1010,0011}, o panaikinant pokytį ketvirtam įėjime, gauname rinkinių porą {1010,0110}. Įėjimo pokyčio panaikinimas gali sąlygoti ir kai kurių pokyčių išnykimą išėjimuose. Jei taip atsitinka, tai galima tvirtinti, kad įėjimas, kuriam buvo panaikintas pokytis įtakoja išėjimą, kuriam pokytis išnyko. Įtaka pasireiškia schemos keliais. Čia galima išskirti keturis atvejus: įėjimo pokyčio 0→1 panaikinimas sąlygoja išėjimo pokyčio 0→1 išnykimą, įėjimo pokyčio 0→1 panaikinimas sąlygoja išėjimo pokyčio 1→0 išnykimą, įėjimo pokyčio 1→0 panaikinimas sąlygoja išėjimo pokyčio 0→1 išnykimą, įėjimo pokyčio 1→0 panaikinimas sąlygoja išėjimo pokyčio 1→0 išnykimą. Kadangi vieno įėjimo pokyčio panaikinimas tiesiogiai sąlygoja pokyčio išnykimą išėjime, tai tokią įtaką pagal analogiją su schemos kelių tikrinimu, vadinsime robust įtaka. Gali būti ir netiesioginis įtakojimas, kai įėjimo pokyčio panaikinimas iššaukia papildomo išėjimo pokyčio atsiradimą. Tai reiškia, kad įėjimo pokytis blokavo kito pokyčio įtaką į išėjimą ir pokytį panaikinus išnyko šis blokavimas, o išėjime atsirado naujas pokytis. Čia taip pat galima išskirti keturis atvejus: įėjimo pokyčio 0→1 panaikinimas sąlygoja papildomo išėjimo pokyčio 0→1 atsiradimą, įėjimo pokyčio 0→1 panaikinimas sąlygoja papildomo išėjimo pokyčio 1→0 atsiradimą, įėjimo pokyčio 1→0 panaikinimas sąlygoja papildomo išėjimo pokyčio 0→1 atsiradimą, įėjimo pokyčio 1→0 panaikinimas sąlygoja papildomo išėjimo pokyčio

1→0 atsiradimą. Netiesioginę įtaką vadinsime nonrobust analogiškai kaip ir transition gedimų tikrinime.

Bendru atveju turime du įėjimo poveikius P1=<p11,p21,p31,…,pi1,…,pn1> ir P2=<p12,p22,p32,…,pi2,…,pn2> ir dvi reakcijas į tuos poveikius R1=<r11,r21,r31,…,rj1,…,rm1> ir R2=<r12,r22,r32,…,rj2,…,rm2>. Įėjimų įtaką į išėjimus galima atvaizduoti matrica ||X||2n x 4m. Matricos elementas x2i-1,4j-3=1, jei pi1=0, pi2=1, rj1=0, rj2=1, ir prilyginus pi2=0 gauname rj2=0. Matricos elementas x2i-1,4j-2=1, jei pi1=0, pi2=1, rj1=1, rj2=0, ir prilyginus pi2=0 gauname rj2=1. Matricos elementas x2i-1,4j-1=1, jei pi1=0, pi2=1, rj1=0, rj2=0, ir prilyginus pi2=0 gauname rj2=1. Matricos elementas x2i-1,4j=1, jei pi1=0, pi2=1, rj1=1, rj2=1, ir prilyginus pi2=0 gauname rj2=0. Matricos elementas x2i,4j-3=1, jei pi1=1, pi2=0, rj1=0, rj2=1, ir prilyginus pi2=1 gauname rj2=0. Matricos elementas x2i,4j-2=1, jei pi1=1, pi2=0, rj1=1, rj2=0, ir prilyginus pi2=1 gauname rj2=1. Matricos elementas x2i,4j-1=1, jei pi1=1, pi2=0, rj1=0, rj2=0, ir prilyginus pi2=1 gauname rj2=1. Matricos elementas x2i,4j=1, jei pi1=1, pi2=0, rj1=1, rj2=1, ir prilyginus pi2=1 gauname rj2=0. Matome, kad visais atvejais pi1 ir pi2 reikšmės yra priešingos ir keičiant pi2 reikšmę turi keistis ir rj2 reikšmė. Įėjimą i matricoje X atitinka dvi eilutės 2(i-1) ir 2i. Eilutė 2(i-1) atitinka įėjimo pokytį 0→1, o eilutė 2i atitinka įėjimo pokytį 1→0. Išėjimą j matricoje X atitinka keturi stulpeliai. Stulpelis 4(j-3) rodo robust įtaką išėjimo pokyčiui 0→1, stulpelis 4(j-2) rodo robust įtaką išėjimo pokyčiui 1→0, stulpelis 4(j-1) rodo nonrobust įtaką išėjimo pokyčiui 0→1, stulpelis 4j rodo nonrobust įtaką išėjimo pokyčiui 1→0.

Paimkime paprastą dviejų įėjimų AND ventilį ir poveikių porą P1=<0,0>, P2=<1,1> su reakcijomis R1=<0>, R2=<1>. Abiejuose įėjimuose ir išėjime yra pokyčiai p1(0→1), p2(0→1), r1(0→1). Įėjimų įtaką į išėjimą vaizduos matrica ||X||4x4. Panaikinus pokytį pirmam įėjime p1(0→0) išnyksta pokytis ir išėjime r1(0→0) ir atžymimas matricos elementas x1,1=1. Analogiškai panaikinus pokytį antram įėjime p2(0→0) išnyksta pokytis ir išėjime r1(0→0) ir atžymimas matricos elementas x3,1=1. Poveikių porai P1=<1,1>, P2=<0,1> su reakcijomis R1=<1>, R2=<0> ir esant pokyčiams p1(1→0), p2(1→1), r1(1→0) atžymimas matricos elementas x2,2=1, o poveikių porai P1=<1,1>, P2=<1,0> su reakcijomis R1=<1>, R2=<0> ir esant pokyčiams p1(1→1), p2(1→0), r1(1→0) atžymimas matricos elementas x4,2=1. Poveikių pora P1=<1,1>, P2=<0,0> su reakcijomis R1=<1>, R2=<0> esant pokyčiams p1(1→0), p2(1→0), r1(1→0) neatžymi jokio matricos elemento, nes panaikinus pokytį bet kuriam įėjime išėjime pokytis neišnyksta. Poveikių poros P1=<0,1>, P2=<1,0> su reakcijomis R1=<0>,

R2=<0> ir P1=<1,0>, P2=<0,1> su reakcijomis R1=<0>, R2=<0>, esant pokyčiams p1(0→1), p2(1→0), r1(0→0) ir p1(1→0), p2(0→1), r1(0→0) nekeičia išėjimo reikšmės tačiau panaikinus pokytį p2(1→0) arba p1(1→0), t. y. padavus signalus (1→1) išėjime atsiranda pokytis r1(0→1). Tai įėjimo įtaka į išėjimą nonrobust tipo ir matricoje X atžymimi elementai x4,3=1 ir x2,3=1. Rezultatai visoms galimoms poveikių poroms parodyti Lentelėje I. Kiekvienai poveikių porai apatinėse eilutėse parodyta kokie pokyčių panaikinimai įtakoja išėjimą ir kokie atžymimi matricos elementai. Matome, kad penkios poveikių poros nenustato įėjimų įtakos į išėjimus. Poveikių pora 3 nustato tą pačią įtaką tarp įėjimų ir išėjimų kaip ir poveikių poros 6 ir 9 kartu.

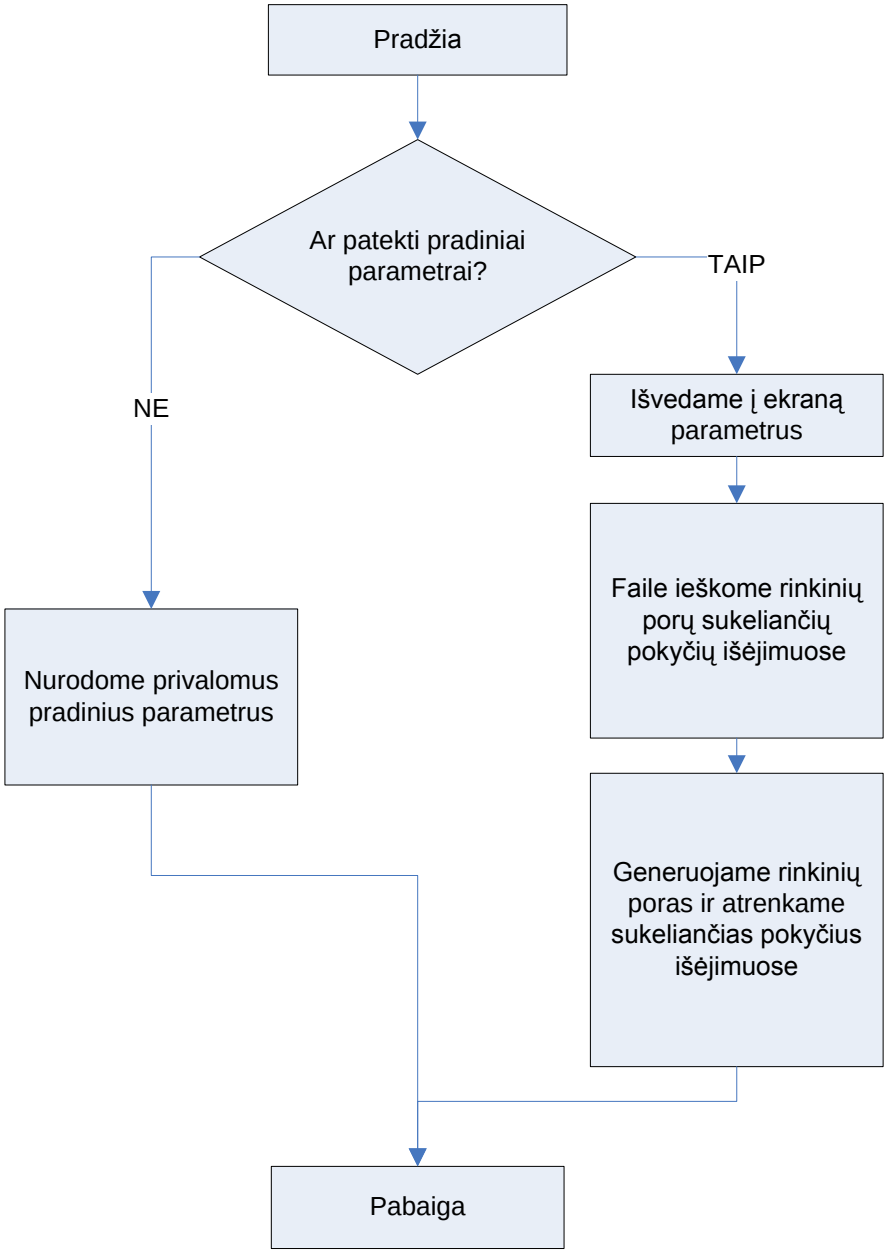
2. Metodas

3.1. Funkcinio vėlinimo testo generavimas

Žinant funkcinio vėlinimo testo kokybės kriterijus galima vykdyti funkcinio testo generavimą. Kai testo generavimo metu naudojamas imitacinis schemos modelis ir nežinoma konkreti schemos realizacija, tenka naudotis poveikio porų atsitiktine paieška. Tai pat žinant kaip galima poveikių porą modifikuoti, kad modifikavimo pasekoje daugiau įėjimų turėtų įtakos į išėjimus (Tam tikslui galima keisti kai kurias fiksuotas įėjimo signalų reikšmes formuojant pokytį (0→1) arba (1→0) arba panaikinti pokytį). Šį pakeitimą tikslinga įvesti tuo atveju, kai jis nustato naujas įtakas į išėjimą ir nekeičia jau nustatytų įėjimų įtakos į išėjimus. Jeigu įvedant pakeitimą prarandamos kai kurios jau nustatytos įtakos į išėjimus, tikslinga turimos poveikių poros nekeisti, bet formuoti naują poveikių porą su kitais pakeitimais. Tokiu būdu išnagrinėję visas poveikių poros įėjimo signalų reikšmes, gausime modifikuotą pirminę poveikių porą ir naujas poveikių poras, nustatančias papildomas įėjimų įtakas į išėjimus.

Schemai, turinčiai n, įėjimų galima sugeneruoti 2^n skirtingų įėjimo poveikių ir 2^n(2^n-1) poveikių porų. Tačiau nemažai poveikių porų įtakos tarp įėjimų ir išėjimų nustatymo požiūriu yra ekvivalentiškos, tai yra nustato tas pačias įėjimų įtakas į išėjimus. Todėl galima apsiriboti atsitiktiniu antro poros poveikio generavimu, o pirmą poveikį pradžioje prilyginti antram ir atlikti poros modifikavimą pagal PPG [3] algoritmą. Kadangi schemos gali turėti iki kelių šimtų ar tūkstančių įėjimų, tai teorinis galimų testinių rinkinių kiekis padidėja keturgubai, pridedant po papildomą įėjimą. Tarkim ISCAS-85 C7552 schema turi 206 įėjimus, tad tokiai schemai galima sugeneruoti 2^206*2^205, o tai jau šiek tiek daugiau nei 5*10^123. Dėl šios priežasties reikia metodo kaip sužinoti, ar duodanti pokytį rinkinių pora nėra jau testuota. Šios problemos sprendimas yra porų pokyčius žymėti X matricoje [3] turinčioje 2n x 2m nariu, kai n yra įėjimų skaičius, o m išėjimų. Tarpinius rezultatus būtina saugoti taip jog jų pakaktu darbo pratęsimui po programos nutraukimo.

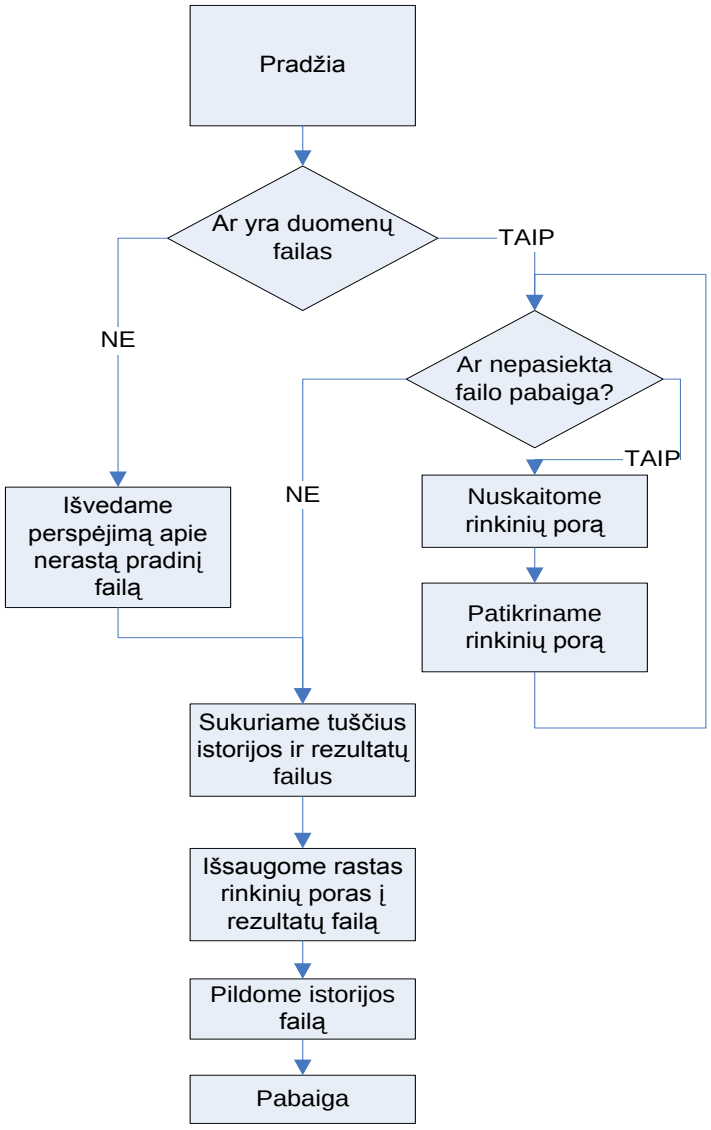
3.2. Realizuojamo algoritmo loginė algoritmo išraiška



3 pav. Loginė programos diagrama

Programos kodas privalo žinoti pradinius duomenis (3 pav.): iteracijų skaičius, schemos įėjimų skaičius, schemos išėjimų skaičius, generavimų kiekis vienoje iteracijoje, bei duomenų ir rezultatų failų vardai. Trūkstant bet kurio parametro gražinamas reikiamų parametrų sąrašas. Gavus tinkamus parametrus programa išveda į ekraną juos ir pradeda duomenų failo apdorojimą.

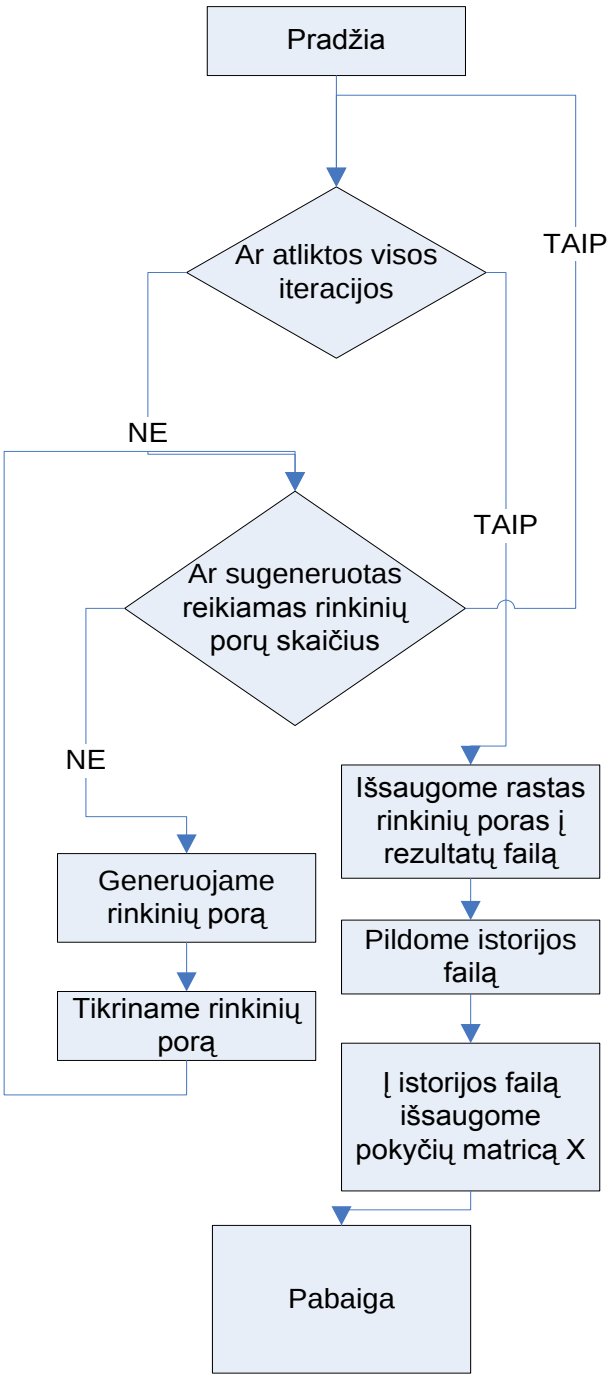
3.3. Duomenų nuskaitymas



4 pav. Duomenų nuskaitymas

Prieš pradedant rinkinių generavimą, reikia nuskaityti ankstesnių generavimų duomenis (4pav.). Neradus duomenų failo grąžinamas klaidos pranešimas, sukuriami tušti istorijos ir duomenų failai ir keliaujama prie duomenų generavimo. Jei duomenų failas rastas, nuskaitoma viena rinkinių pora, tada ji patikrinama ar tinkama, jei tinkama išsaugoma atmintyje, ir keliaujama prie sekančios poros. Ciklas kartojasi tol kol nuskaitomas visas duomenų failas, po to rasti tinkami rinkiniai saugomi rezultatu faile, apie tai pažymint istorijos faile.

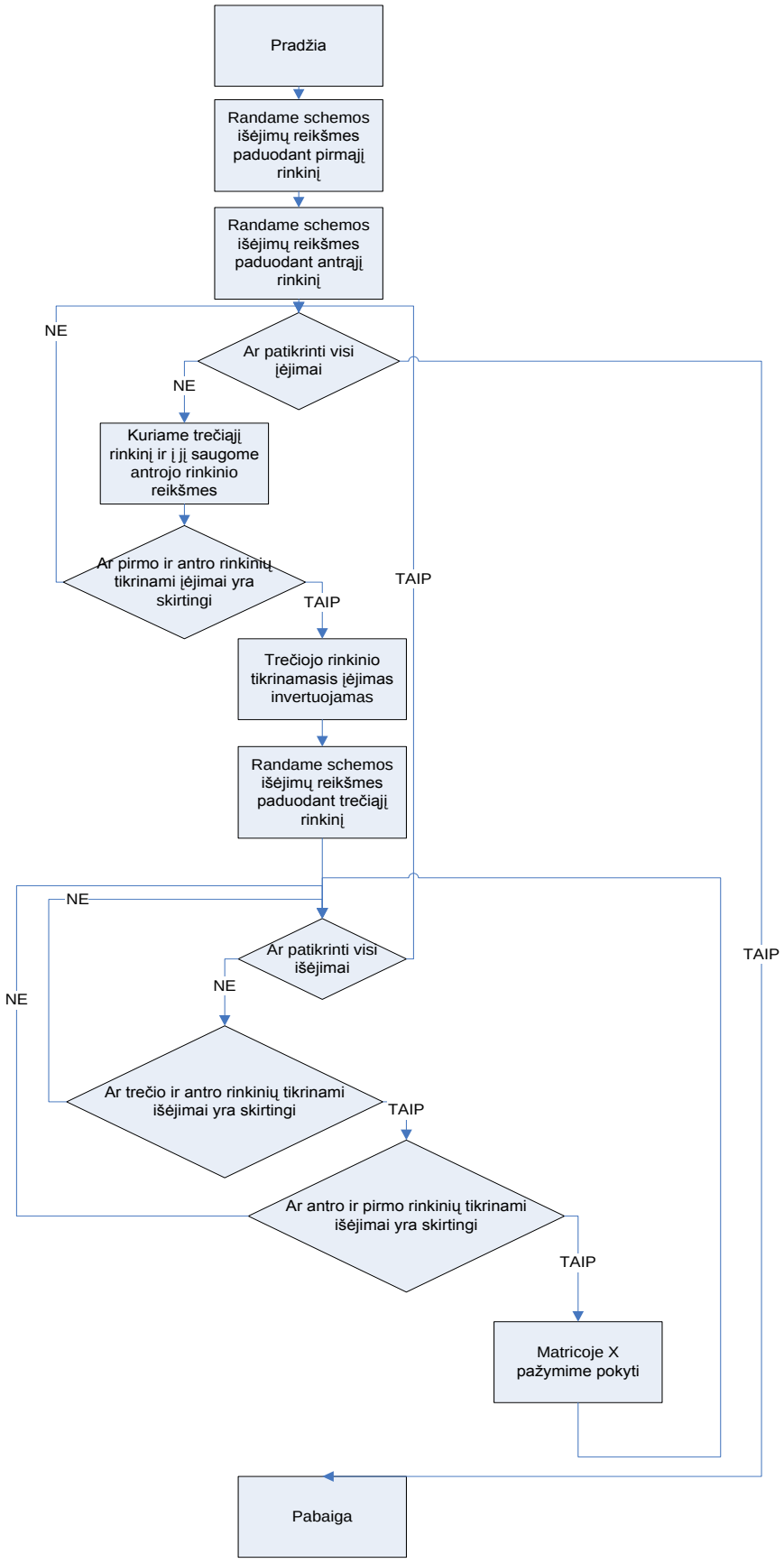
3.4. Rinkinių generavimas



5 pav. Generatoriaus loginė diagrama

Vartotojas pats nurodo reikiama iteracijų ir generavimų kiekį vienoje iteracijoje. Pradžioje tikrinamas ar pasiektas maksimalus iteracijų skaičius (5 pav.) ir ar pasiektas generavimų skaičius iteracijoje. Kiekvieno generavimo metu sugeneruojama ir patikrinama tik viena rinkinių pora, taip taupomi atminties resursai.

3.5. Rinkinių tikrinimas



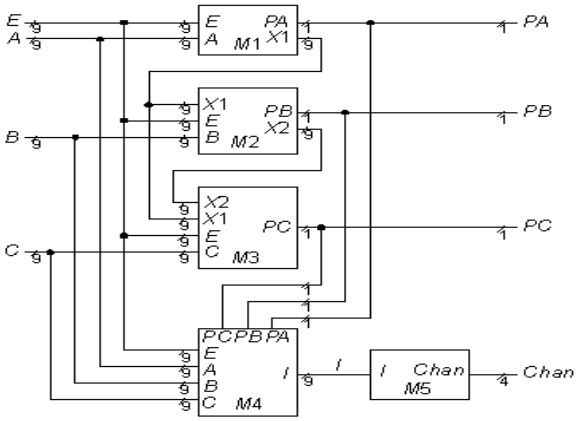
6 pav. Rinkinių tikrinimo diagrama

Viena iš svarbiausių algoritmo dalių tai testinio rinkinio tikrinimas (6 pav.). Pradžioje randamos schemos išėjimų reikšmės prie abiejų rinkinio įėjimo reikšmių. Toliau dirbama su kiekviena įėjimo reikšme. Kuriamas trečias įėjimų rinkinys kuriame išsaugomos antrojo įėjimo reikšmės. Tada tikrinama ar antrojo ir trečiojo rinkinių tikrinamos reikšmės yra lygios. Jei reikšmės lygios, o tai reiškias jog nėra jokio pokyčio, einama prie sekančio įėjimo. Jei įėjimų reikšmės skiriasi, tai trečiojo rinkinio aktyvusis įėjimas yra invertuojamas ir randamos išėjimo reikšmės padavus trečiąjį reikšmių rinkinį. Toliau su šiais trim įėjimų rinkiniais tikrinami visi išėjimai. Paimamas pirmas išėjimo narys ir tikrinama ar trečio ir antro įėjimų rinkinių pora duoda pokyti išėjime ir jei pokyčio nėra tai keliaujama prie kito išėjimo. Jei antro ir trečio rinkinių pora duoda pokyti pasirinktame išėjime, tikrinama ar duoda pokyti antro ir pirmo rinkinių pora. Jei ir antro ir pirmo rinkinių pora duoda pokyti, tai X matricoje atžymimas pokytis K ir keliaujama prie sekančio išėjimo. Bet kuris rinkinys sugeneravęs atžymą X matricoje saugomas.

3. Tyrimo rezultatai ir jų įvertinimas

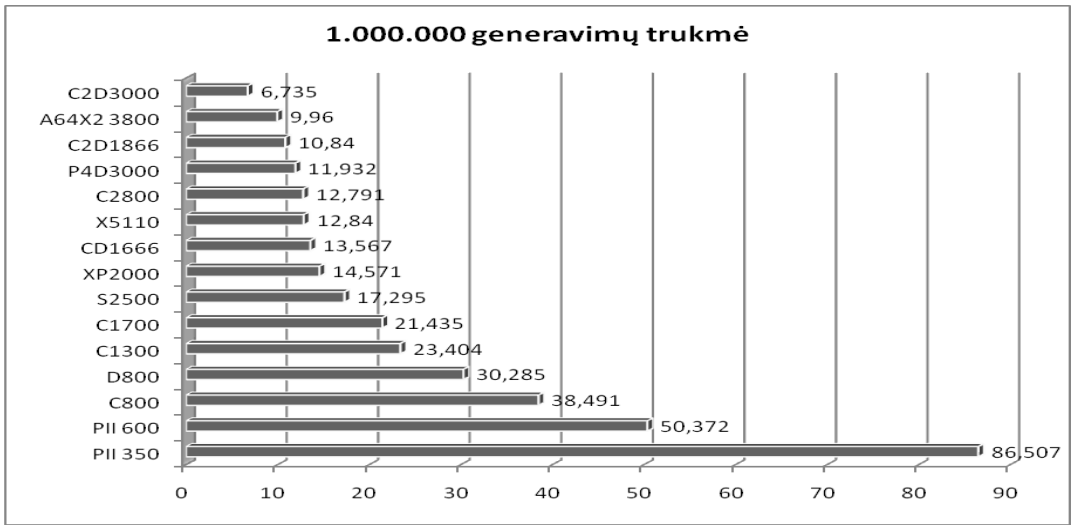
4.1. ISCAS-85 c432

Statistika: 36 įėjimai; 7 išėjimai; 160 ventilių (4 AND + 119 NAND + 19 NOR + 18 XOR); 40 inverterių

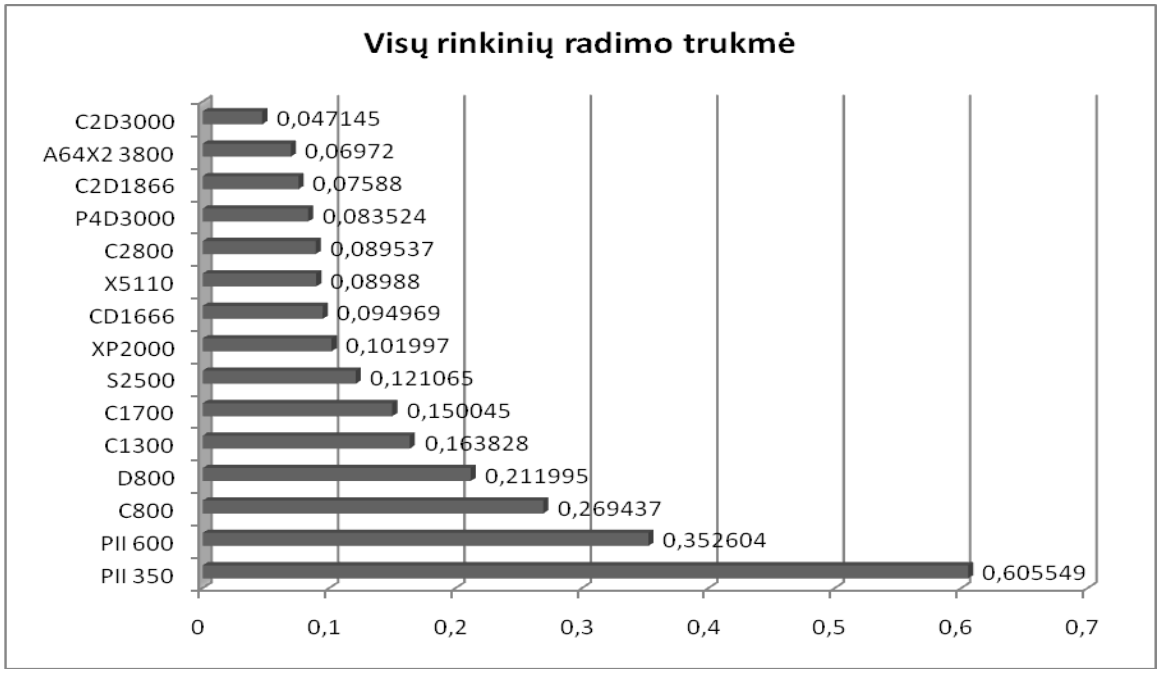


7 pav. c432

c432 (7pav.) yra 27 kanalų pertraukties valdiklis. Įėjimo kanalai sugrupuoti į tris 9-bitų magistrales (A,B ir C), kur bito poziciją kiekvienoje magistralėje nusako pertraukties užklausos prioritetą. Ketvirta 9-bitų magistralė (E) suteikia arba pašalina pertraukties teisę priklausomai nu bitų pozicijos. Septyni išėjimai PA, PB, PC ir Chan[3:0] nurodo, kuris kanalas yra gavęs pertraukties užklausas. Tik tai kanalas, turintis didžiausią prioritetą užklausų magistralėje, gali gauti pertraukties užklausą.

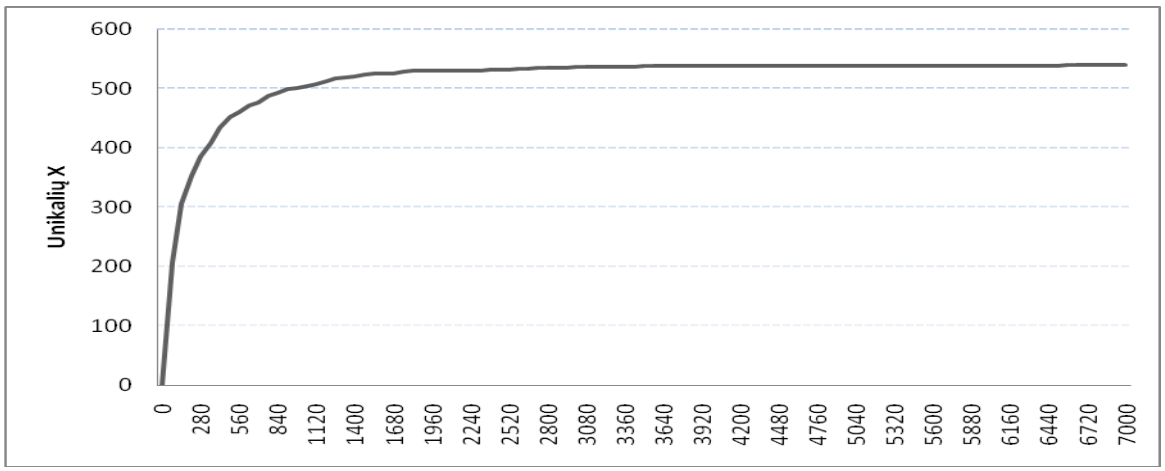


8 pav. c432 1.000.000 generavimų greitaveika



9 pav. c432 7.000 generavimų greitaveika

Ši schema viena iš mažiausiai turinčių ventilių (tik 160), tad schemos tikrinimas nereikalauja didelių resursų, o milijono generavimų laikas (8 pav.) priklausomai nuo pasirinkto procesoriaus svyruoja nuo 6,7 sekundes iki 86,5 sekundes. Žinoma,[25] jog reikia rasti 540 matricos X K elementų, juos pasirinktais procesoriais galima rasti greičiau nei per vieną sekundę (9 pav.) ir nesvarbu kuris procesorius bus pasirinktas.

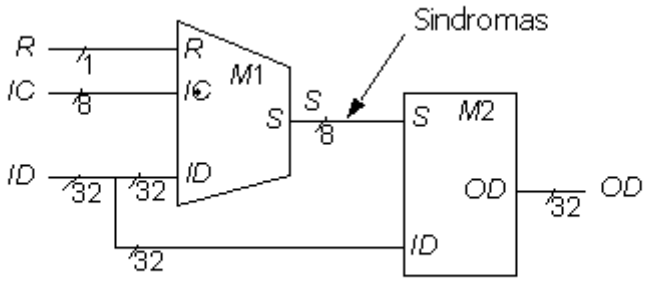


10 pav. c432 Rastų K narių priklausomybė nuo generavimų skaičiaus

Rastų X matricos narių išsibarstymas ganėtinai didelis (10 pav.): po 5% generavimu rasta šiek tiek daugiau nei 75%, po 10% generavimų rasta 88% ieškomų matricos X narių, o po 20% generavimų jau 96%. 80% skaičiavimo laiko sunaudojama rasti 4% reikiamų narių.

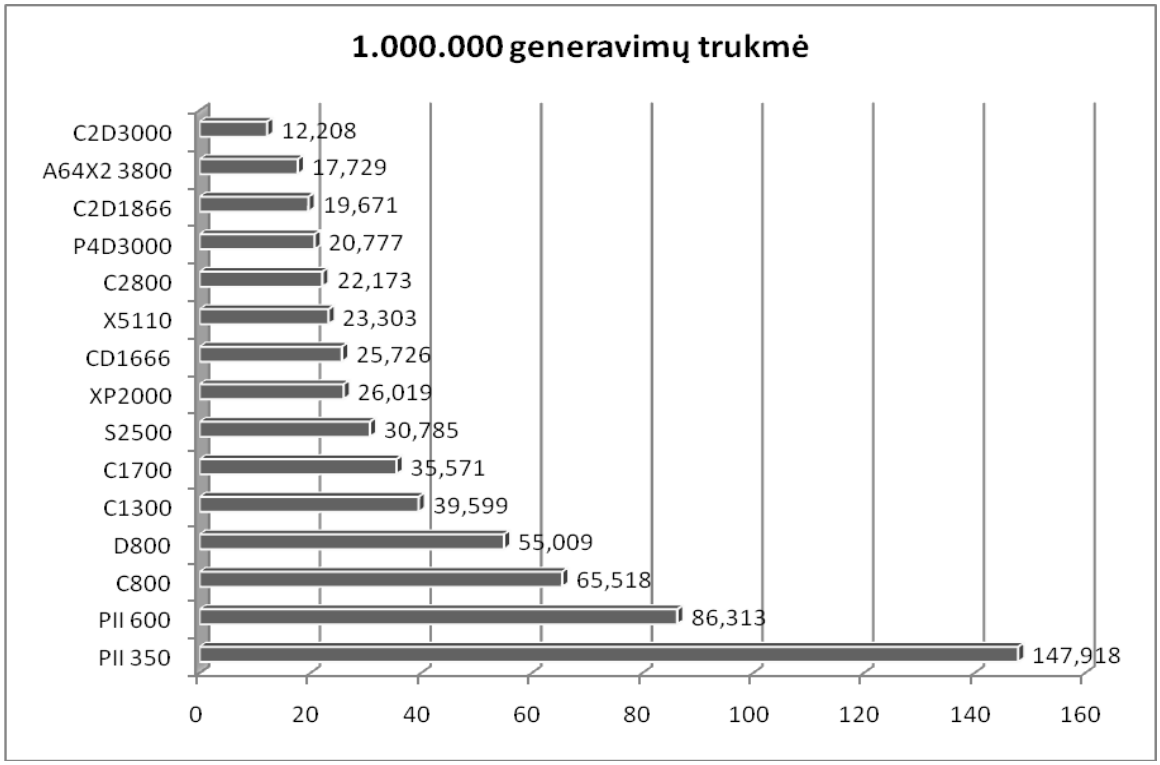
4.2. ISCAS-85 c499

Statistika: 41 įėjimas; 32 išėjimai; 202 ventiliai (56 AND + 40 NAND + 2 OR + 104 XOR); 40 inverterių

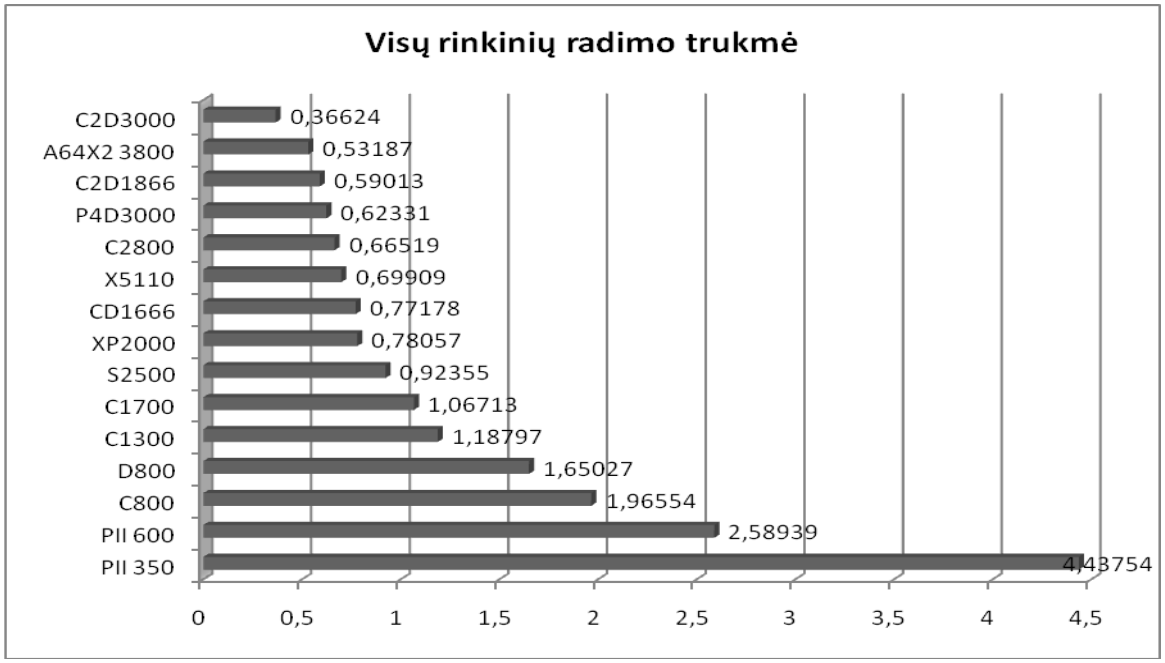


11 pav. c499

ISCAS-85 c499 yra 32-bitų vienos klaidos taisymo grandinė (11 pav.): 41 įėjimas susideda iš 8-bitų vidinės magistralės S, kuri toliau susijungia su 32 pirminiais įėjimais kurie kartu ir 32 pirminiai išėjimai. M2 modulis turi reikalinga korekcijos logika, dėl to c499 gali ištaisyti vieno bito klaidas. S linijos yra suformuotos taip kad generuotu unikalų sindromą kiekvienai įėjimo klaidai.

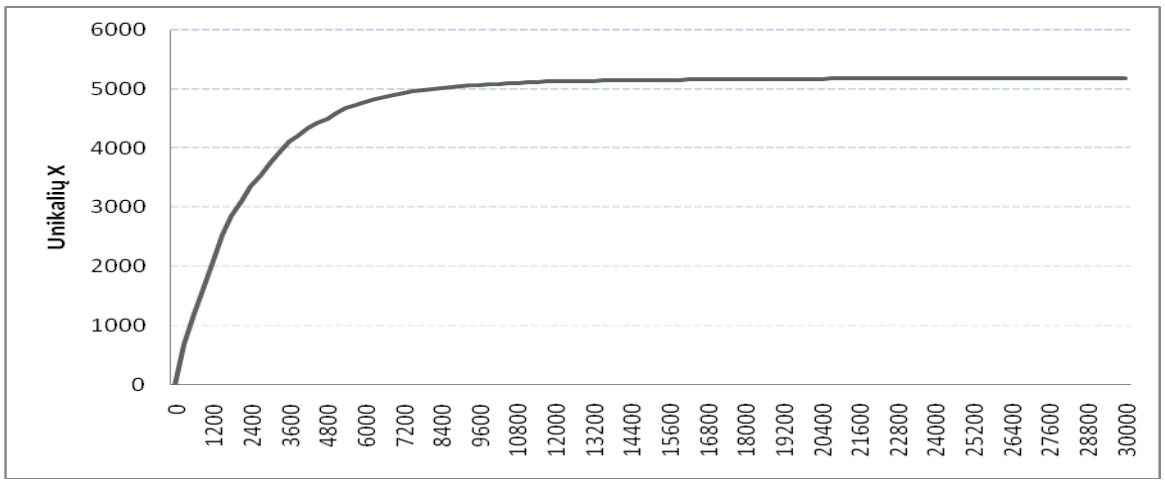


12 pav. c499 1.000.000 generavimų greitaveika



13 pav. c499 30000 generavimų greitaveika

c499 lyginant su c432 turi tik 42 ventiliais didenė ir turi 41 daugiau išėjimą. Nepaisant to, jog ventilių ir įėjimų skaičius padidėjo neženkliai, tačiau jos skaičiavimo greitaveika sulėtėjo dvigubai (12 pav.), o K reikšmių reikia rasti net dešimt kart daugiau, tad visų 5184 K radimas užtrunka beveik aštuonis kart ilgiau (13 pav.)

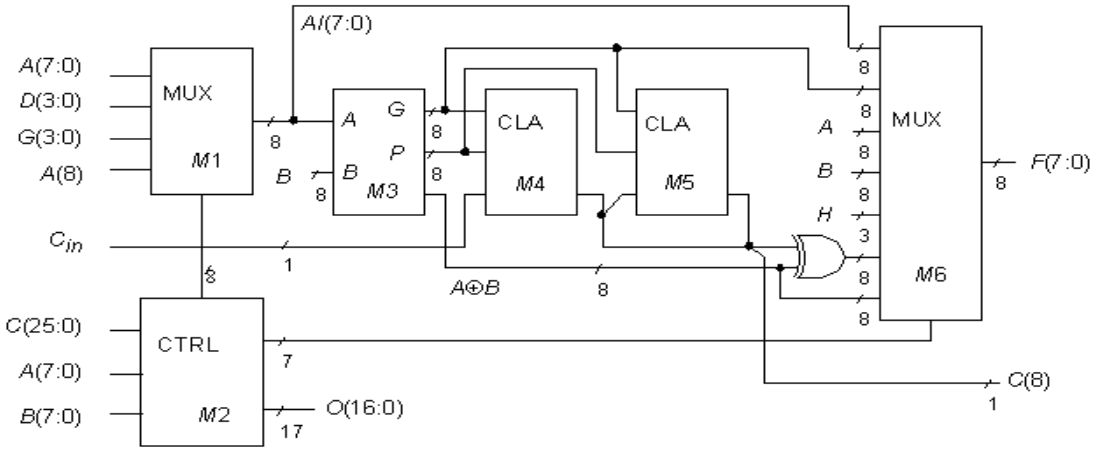


14 pav. c499 Rastų K narių priklausomybė nuo generavimų skaičiaus

Šiai schmai randamų K išsibarstymas vienas iš dižiausių iš visų nagrinejamu 10 ISCAS'85 schemų (14pav.). Po 5% generavimų rasta tik apie 50% visų ieškomų K. Po 20% generavimų rasta 92%, o praejus 30% generavimų rastau jau 97,5% ieškomų K.

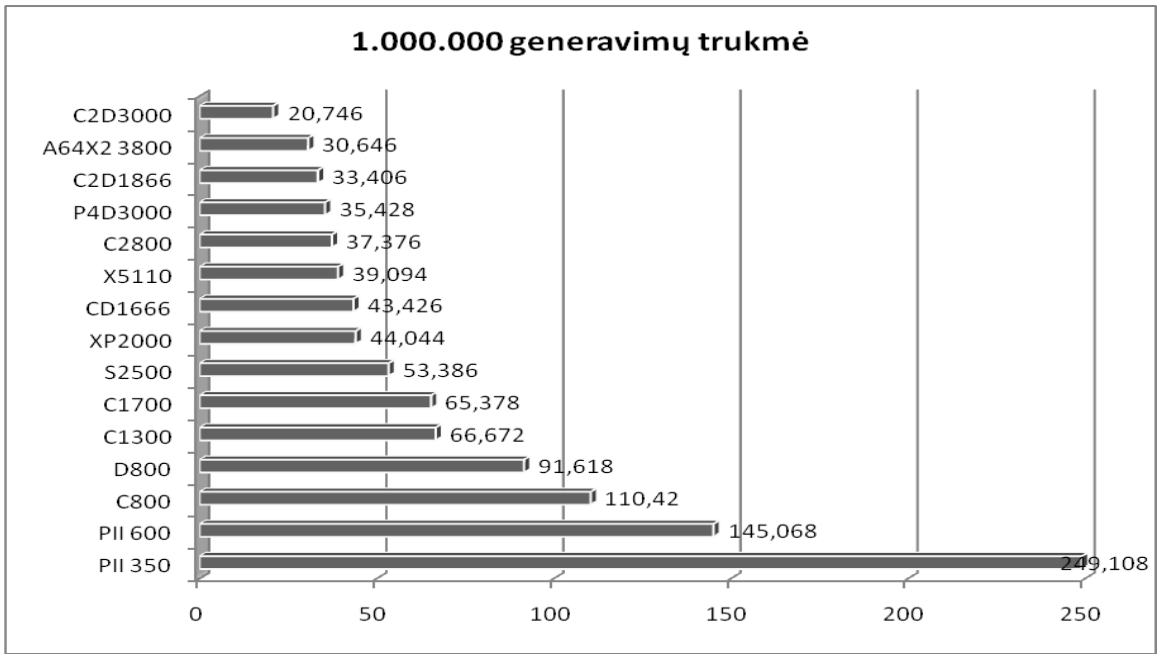
4.3. ISCAS-85 c880

Statistika: 60 įėjimų; 26 išėjimai; 383 ventiliai (143 AND + 150 NAND + 29 OR + 61 NOR); 63 inverteriai; 26 buferiai

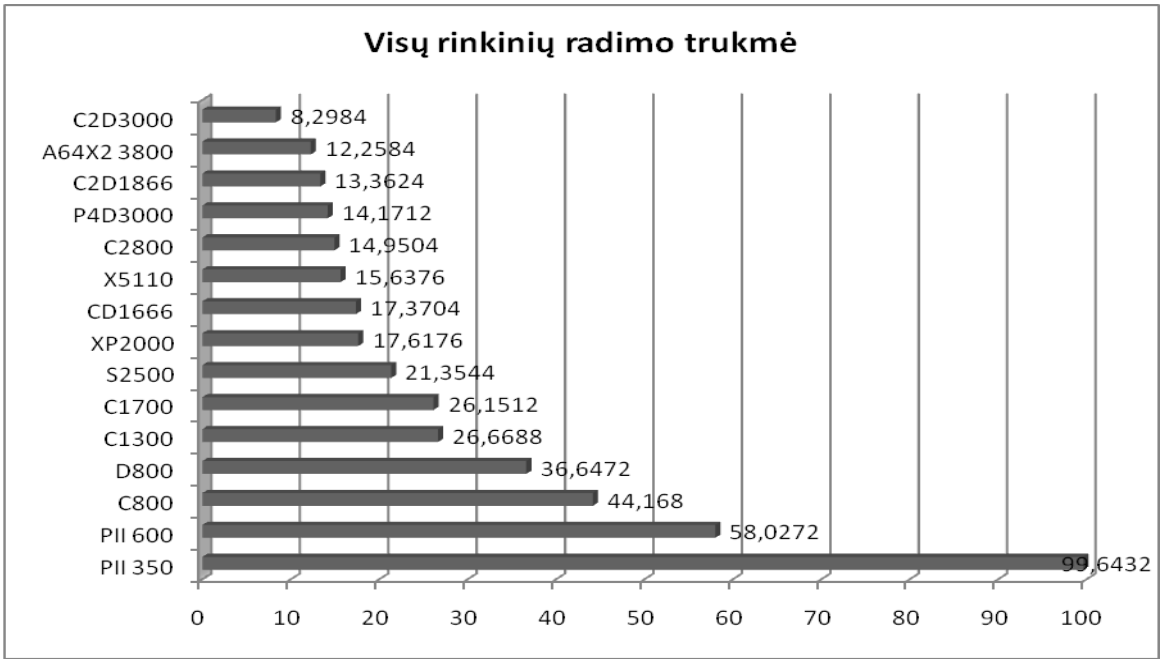


15 pav. c880

ISCAS-85 c880 yra 8-bitų ALU kurio aukštesnio lygio modelis parodytas 15 paveiksle. Šios 8-bitų ALU schemos pagrindas yra 74283-stiliaus sumatorius. M1 ir M6 multiplekseriai. Yra valdomi M2 modulio paremto vertikaliu mikrokodu, t.y. naudojant schema reikia jog tik viena funkcija būtu aktyvi C(25:0) įėjime.

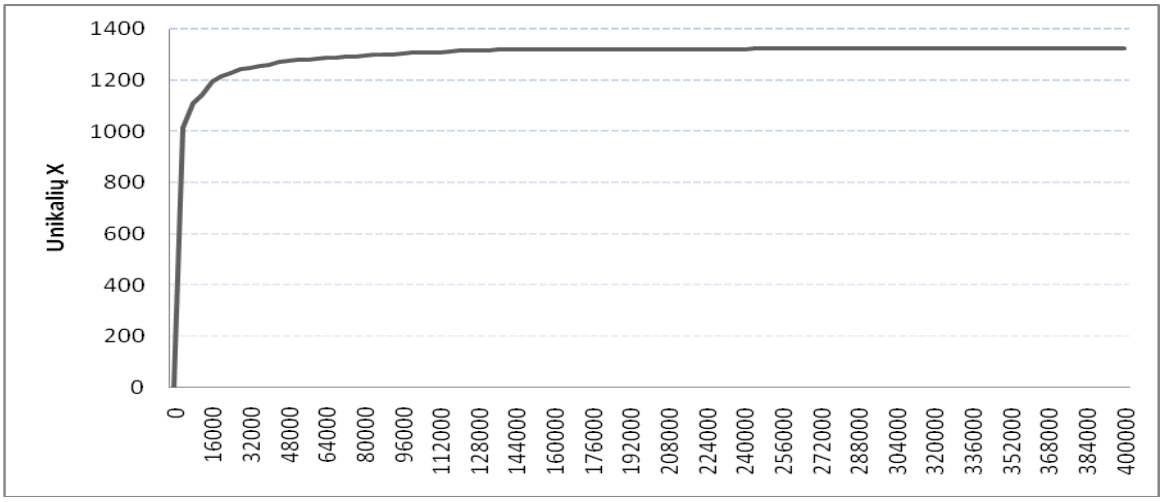


16 pav. c880 1.000.000 generavimų greitaveika



17 pav. c880 400000 generavimų greitaveika

Ši schema turi net 60 iėjimų, už ją daugiau turi tik c3540, tačiau tai generavimų greitaveikos smarkiai neitakoja (16 pav.), o 400.000 generavimų neužima daug laiko nuo 8 iki 100 sekundžių priklausomai nuo procesoriaus galingumo (17 pav).



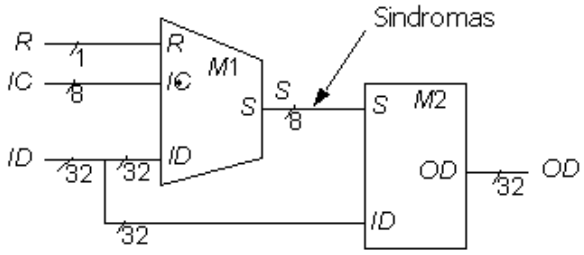
18 pav. c880 rastų K narių priklausomybė nuo generavimų skaičiaus

Didėjant reikiamų generavimų skaičiui K išsibarstymas pagal generavimų kiekį tankėja generavimų pradžioje. Jau po 1% generavimų randama77% ieškomų K (18 pav.). Po 10% generavimu rasta net 95% ieškomų K.

4.3. ISCAS-85 c1355

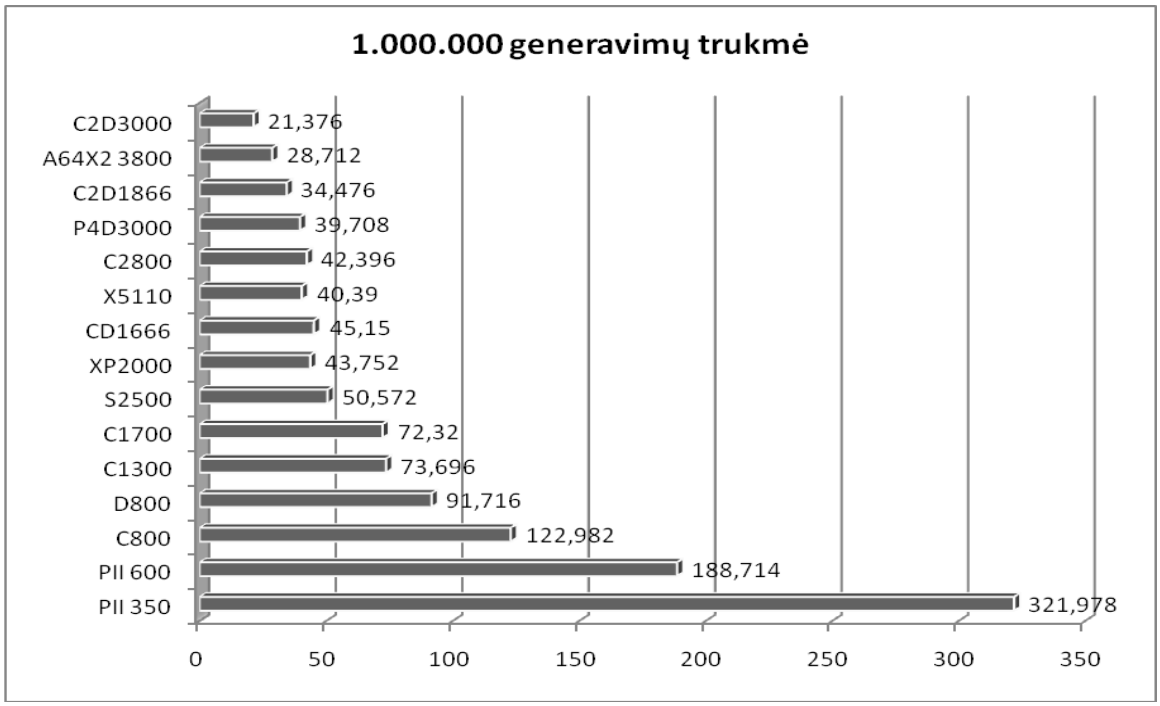
Statistika: 41 įėjimas; 32 išėjimai; 546 ventiliai (88 AND + 456 NAND + 2 OR); 40 inverterių; 32 buferiai

ISCAS-85 c1355 yra 32-bitų vienos klaidos taisymo grandinė:

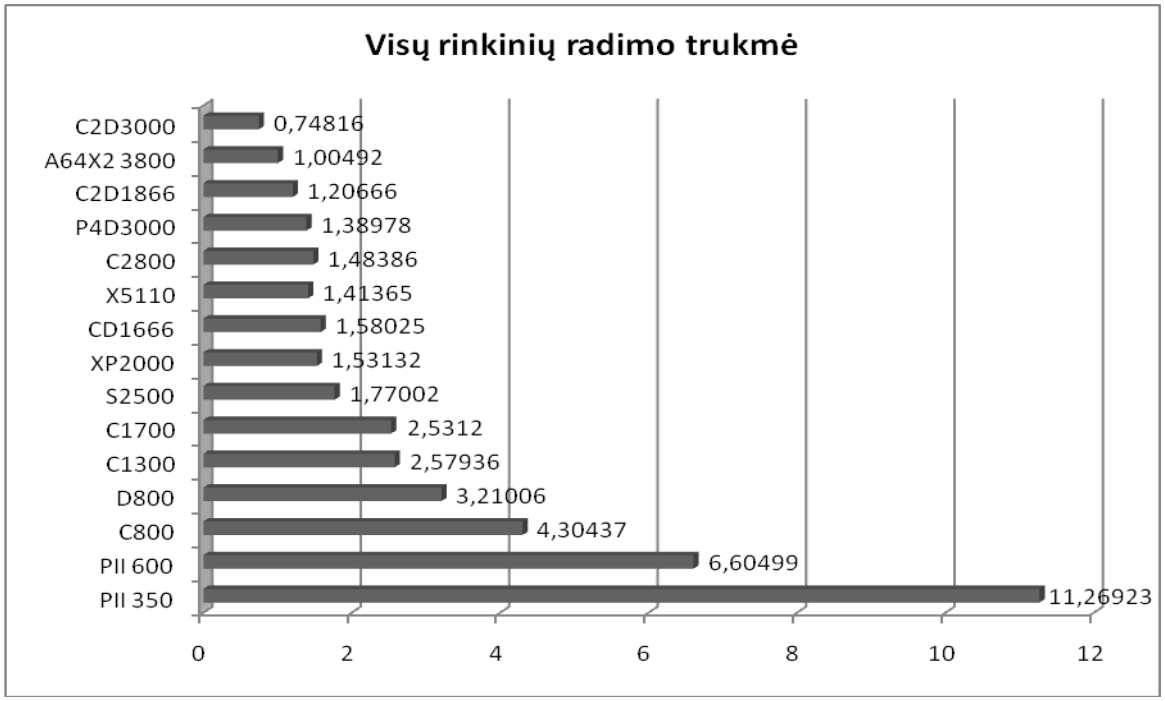


19 pav. c1355

41 įėjimas susideda iš 8-bitų vidinės magistralės S, kuri toliau pasijungia su 32 pirminiais įėjimais kurie kartu ir 32 pirminiai išėjimai. M2 modulis turi reikalinga korekcijos logika, dėl to C1355 gali ištaisyti vieno bito klaidas. S linijos yra suformuotos taip, kad generuotu unikalų sindromą kiekvienai įėjimo klaidai. C1355 veikimas atitinka ISCAS-85 C499, tačiau čia XOR elementai naudoti C499 pakeiti i keturių NAND ventilių atitikmenis.

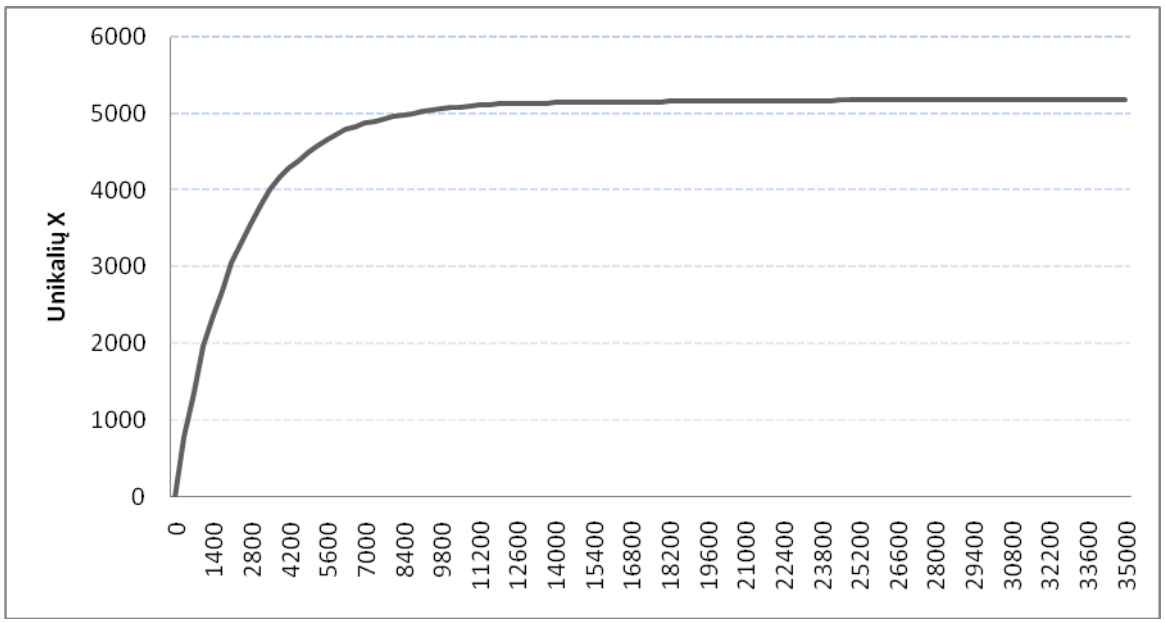


20 pav. c1355 1.000.000 generavimų greitaveika



21 pav. c1355 35000 generavimų greitaveika

c1355 ir turi tą pačią logiką kaip ir c499, tačiau XOR elemnetų pakeitimas 4 NAND elementais turi greitaveikos itakai. C1355 tikrinama beveik du kartus lėčiau (20 pav.) . Analogiškai visų ieškomų K radimas pailgėja ir beveik du kartus (21 pav.)



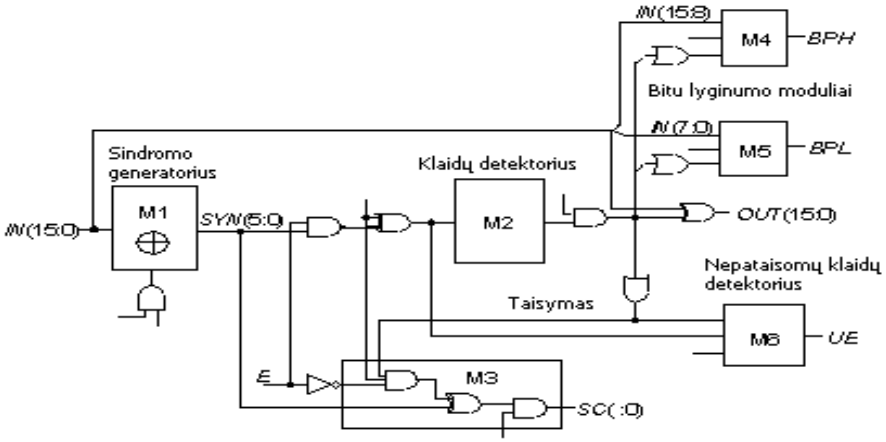
22 pav. c1355 Rastų K narių priklausomybė nuo generavimų skaičiaus

Jei vidinių elementų pakeitimas turėjo įtakos greitaveikai, tai K narių išsibarstymas generavimų intervale lyginant su c499, beveik nepakito. Po 5% generavimų rasta 51% ieškomų K, po 20% generavimų rasta 93% ieškomų K.

4.4. ISCAS-85 c1908

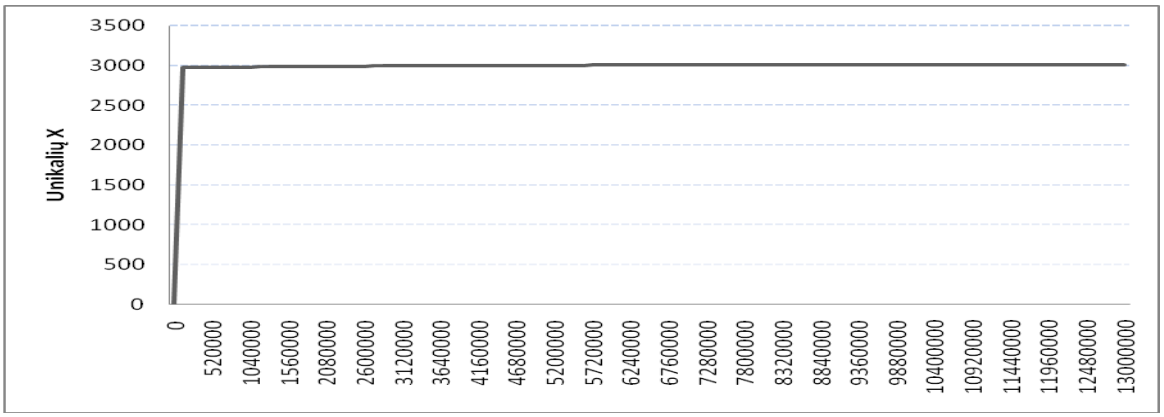
Statistika: 33 įėjimas; 25 išėjimai; 880 ventilių (225 AND + 654 NAND + 1 NOR); 277 inverteriai; 162 buferiai

ISCAS-85 c1908 yra 16-bitų klaidų detektorius/korektorius:



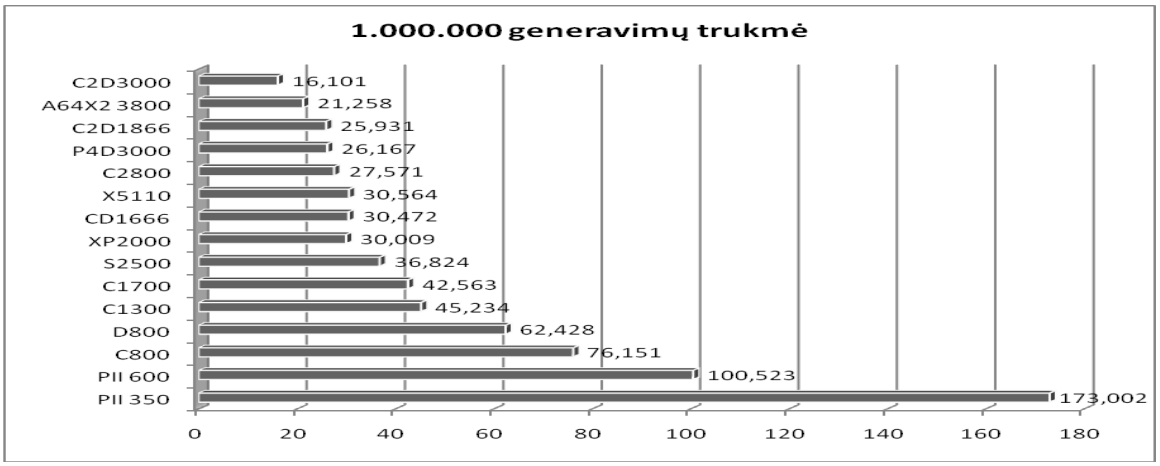
23 pav. c1908

c1908 yra 16-bitų vienos klaidos taisymo ir dvigubos klaidos nustatymo schema. Ji generuoja 6-bitų sindromą iš 16-bitų įėjimo IN, kuris yra ižkoduotas, kad atrasti bito klaidą, jei tokia yra. Jei nustatoma klaida ir valdymo įėjimai yra atitinkamai nustatyti, tada atliekamas klaidos taisymas. C1908 turi išėjimą nusakanti ar nėra nepataisomų klaidų. Schema taip pat gali generuoti sindromo bitus kurie išvedami SC kanalais. Išoriniai sindromo kanalai įgalina apjungti kelias C1908 schemas taip, jog jos galėtų tikrinti ir taisyti klaidas didesniems nei 16-bitų pločio signalams. Ši schema yra labai panaši į AMD (Advanced Micro Devices) Am2960 16-bitų klaidų radimo ir taisymo įrenginį.



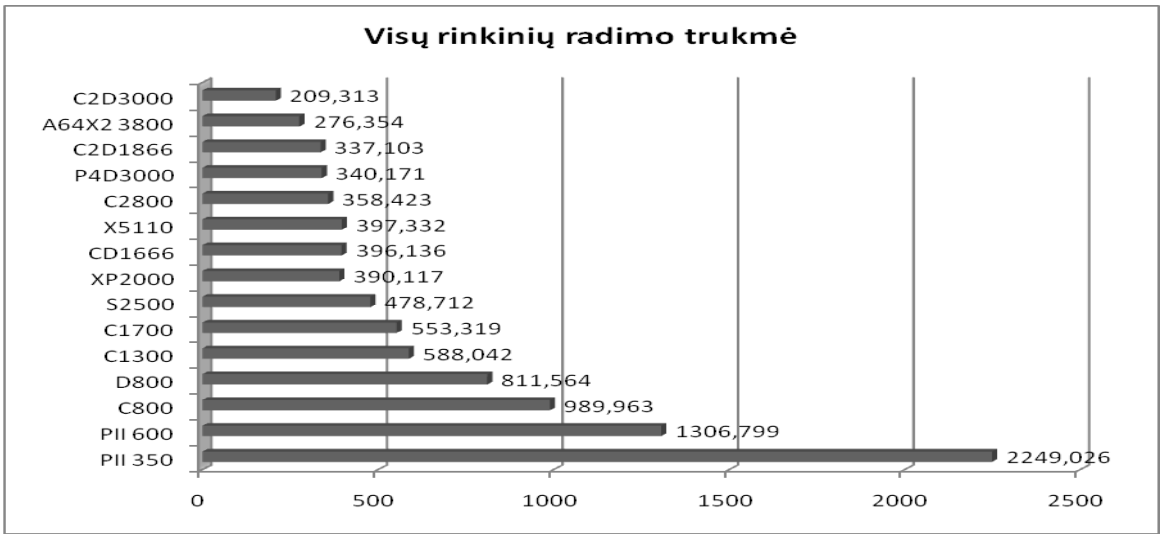
24 pav. c1908 Rastų K narių priklausomybė nuo generavimų skaičiaus

Kad rasti schemos c1908 visus ieškomus 3004 matricos X elementus K reikėjo įvykdyti net 13.000.000 generavimų (24 pav.). Tai įtakojo labai didelį tankį rastų K genrevimo pradžioje. Jau po 1% genreavimų rasta 2973 K o tai yra beveik 99% visų ieškomų K. Po 50% generavimų jau rasta 3000 K, taigi likusius 50% generavimų iškoma likusių 4 K reikšmių.



25 pav. c1908 1.000.000 generavimų greitaveika

c1908 generavimo greitavieka ganėtinai didelė, už ją greičiau generuojamos tik c499 ir c432 (25 pav.) 1.000.000 generavimo laikas svyruoja nuo 16 iki 173 sekundžių.

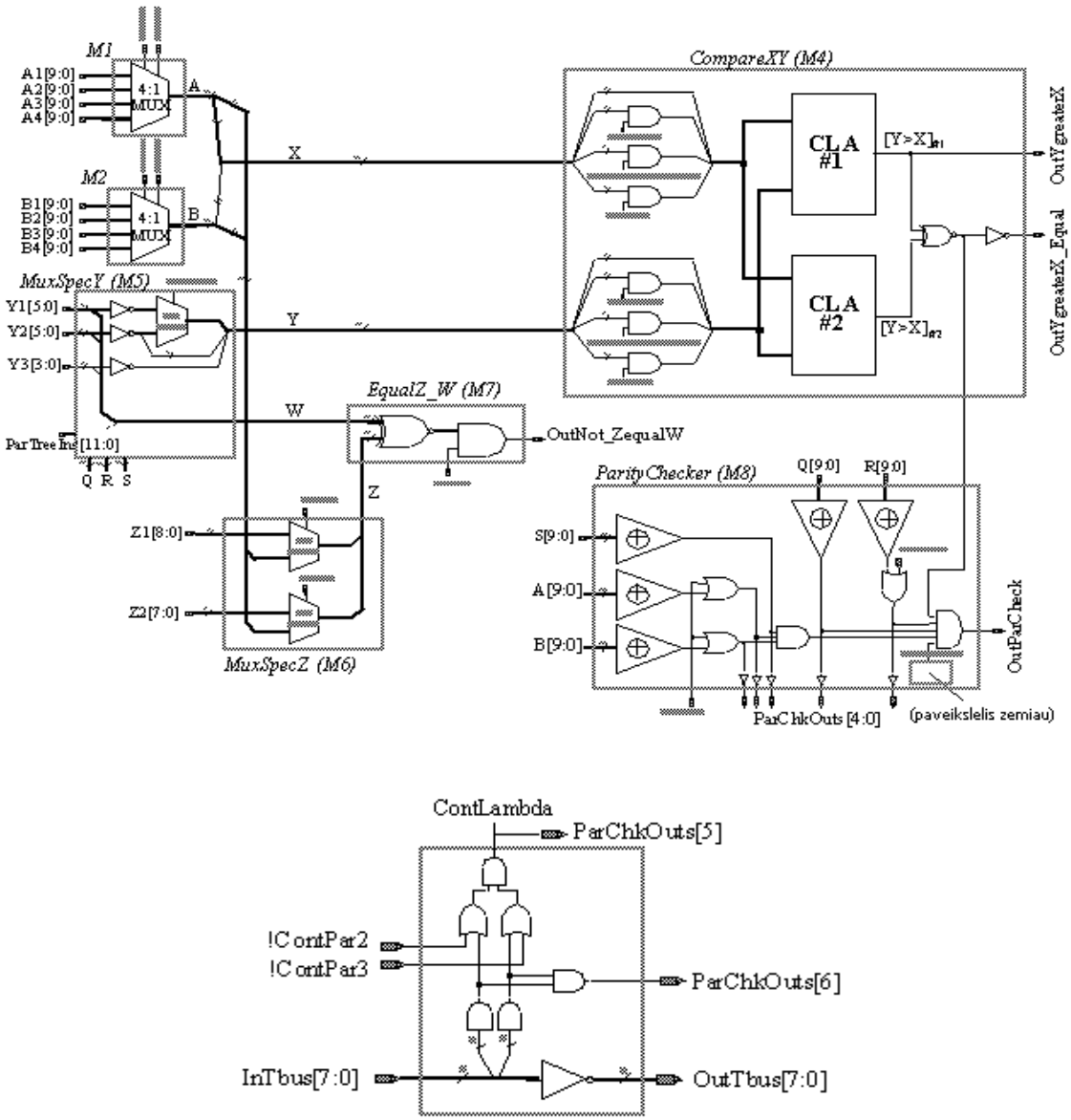


26 pav. c1908 13000000 generavimų greitaveika

Nors schemos generavimų sparta ir yra ganėtinai didelė, tačiau reikalingas 13.000.000 generavimų skaičius prailgina visų ieškomų K generavimo laiką, kuris svyruoja nuo 209s iki 26,2981 min priklausomai nuo pasirinkto procesoriaus.

4.5. ISCAS-85 c2670

Statistika: 233 įėjimai; 140 išėjimų; 1193 ventiliai (529 AND + 575 NAND + 77 OR + 12 NOR); 321 inverteris; 196 buferiai

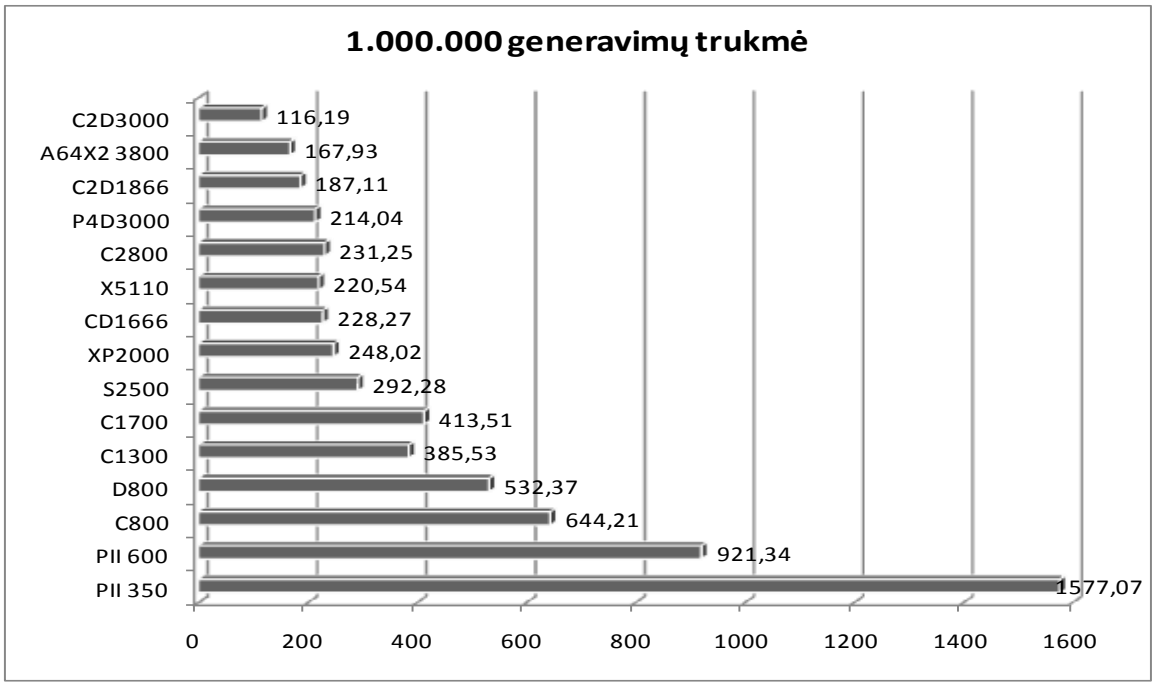


27 pav. c2670

ISCAS-85 c2670 yra 12-bitų ALU turintis komparatorių, lygybės tikrintoja, bei keletą lyginumo šakų. Komparatorius turi du 12-bitų įėjimus Z ir Y, ir apdoroja X>Y naudojant paspartintosios pernašos sudėtuvas (CLA) kurie vykdo !X+Y sumavimus. Taip pat, jis gali būti užprogramuotas kaip 4,6,8 ar 12-bitų tikrintojas. Įdomu tai, jog komparatorius naudoja du identiškus CLA kas dažnai naudojama dubliavimo technologijoje, plačiai naudojamoje klaidoms atspariose sistemose. CLA turi ganėtinai

standartinę struktūra susidedančią iš 3, 4 ir 5-bitų blokų. „*OutYgreaterX_Equal“* konstanta lygi loginiam 1 jei abu CLA išėjimai yra identiški, kas paprastai ir yra. Jei CLA gražina skirtingas reikšmes tada „*OutYgreaterX_Equal“* įgyja loginio 0 reikšmę ir parodo grandinės klaidą. Tai gali nutikti, jei gamybos metu atsirado defektų viename ar abejuos CLA.

M7 modulis atlieka dviejų 17-bitų magistralių lyginimą. M8 modulis sudarytas iš 10 įėjimo lyginumo grandinių, kurių visi išėjimai yra sujungti IR elementais. Jis turbūt atlieka schemos įėjimų būklės tikrinimą. Taip pat schemoje yra keletas mažų gabalėlių turinčių įvairią logiką.

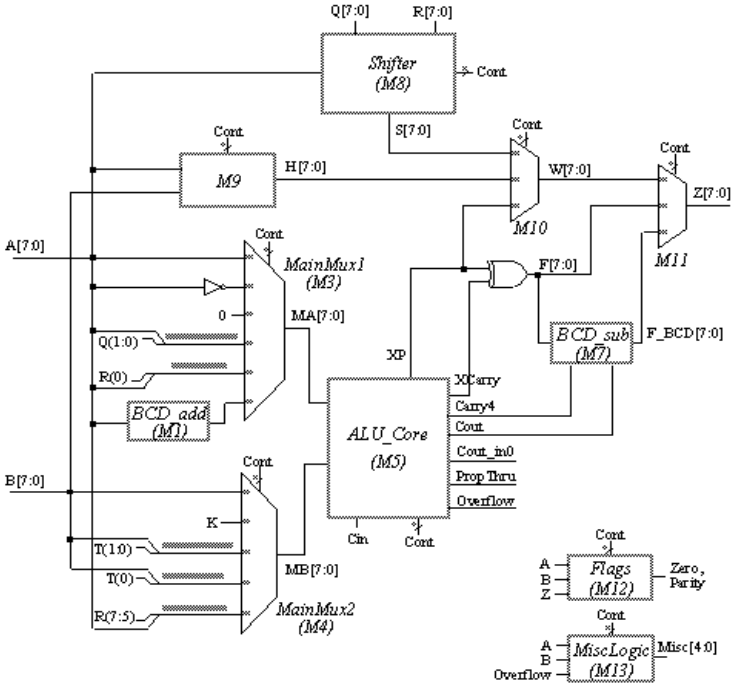


28 pav. c2670 1.000.000 generavimų greitaveika

Nors c2670 generavimo greitaveika nėra pati lėčiausia (28pav.) tačiau iš ieškomų 3320 K rasta tik 2904 K reikšmės. Po pirmo 10.000.000 generavimo rasta 2904 K reikšmės likę 1.500.000.000 generavimų nerado nė vieno unikalaus matricos X elemento K. Tiek generavimų ir jiems sugaišto laiko pakanka surasti visoms aštuonioms: c432, c499, c880, c1355, c1908, c3540, c5315, c6288 rasti visus jų unikalius K ir nesvarbu, jog kitų schemų ir ventilių ir įėjimų skaičius didesnis nei šios schemos.

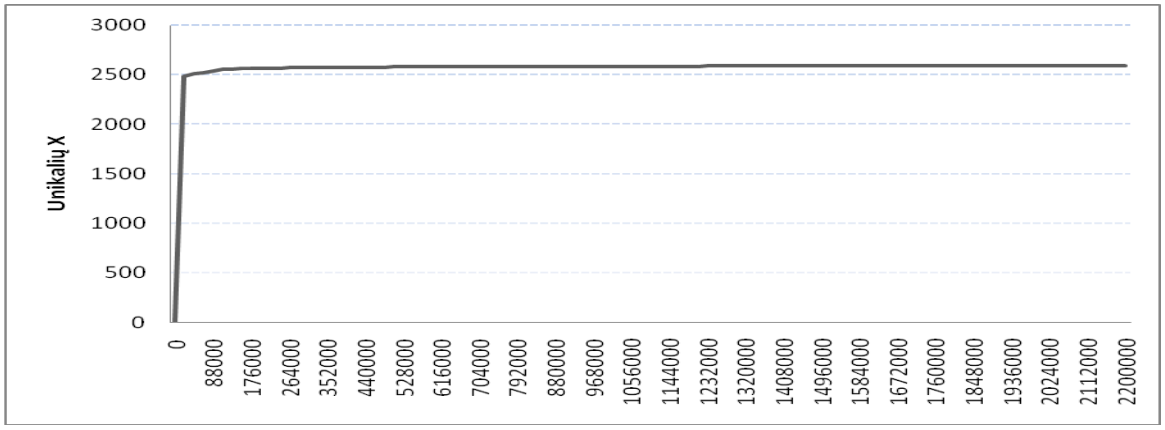
4.7. ISCAS-85 c3540

Statistika: 50 įėjimų; 22 išėjimai; 1669 ventiliai (721 AND + 788 NAND + 92 OR + 68 NOR); 490 inverterių; 223 buferiai



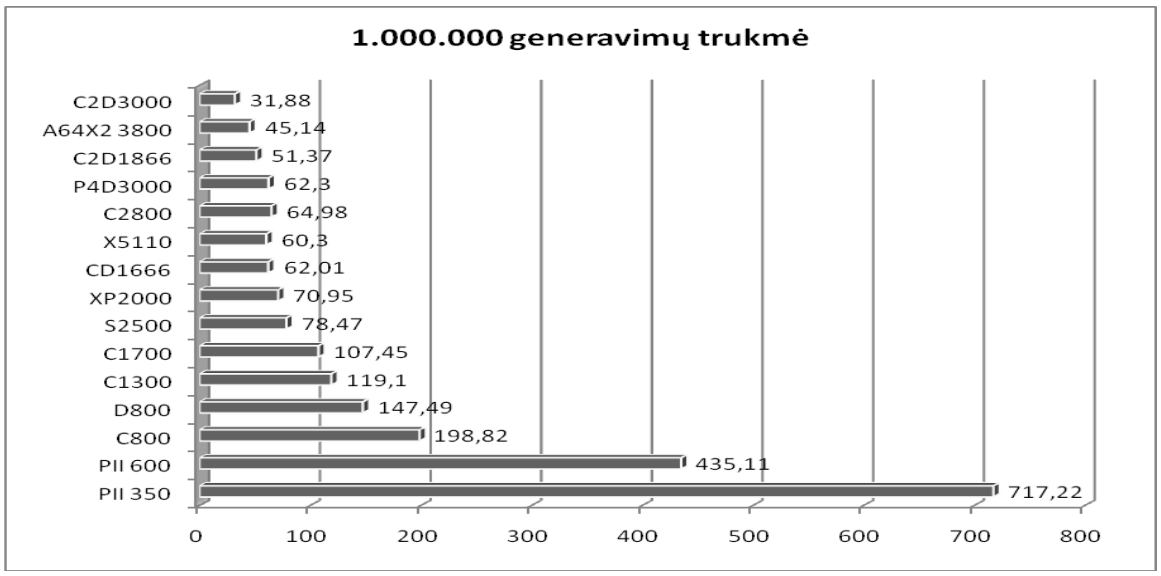
29 pav. c3540

ISCAS-85 c3540 yra 8-bitų ALU galintis atlikti binarinės ir BCD aritmetikos operacijas, o taip pat logines ir postūmio operacijas. Loginės operacijos apjungtos su aritmetinėmis panašiai kaip ir TTL74181 schemoje. BCD priedai įgyvendinti dviejų sudėtinių sumatorių pridedant 6 prie abiejų pirmų operandų skaitmenų ir tada atimant 6 iš rezultato jei negeneruojamas pernešimas.



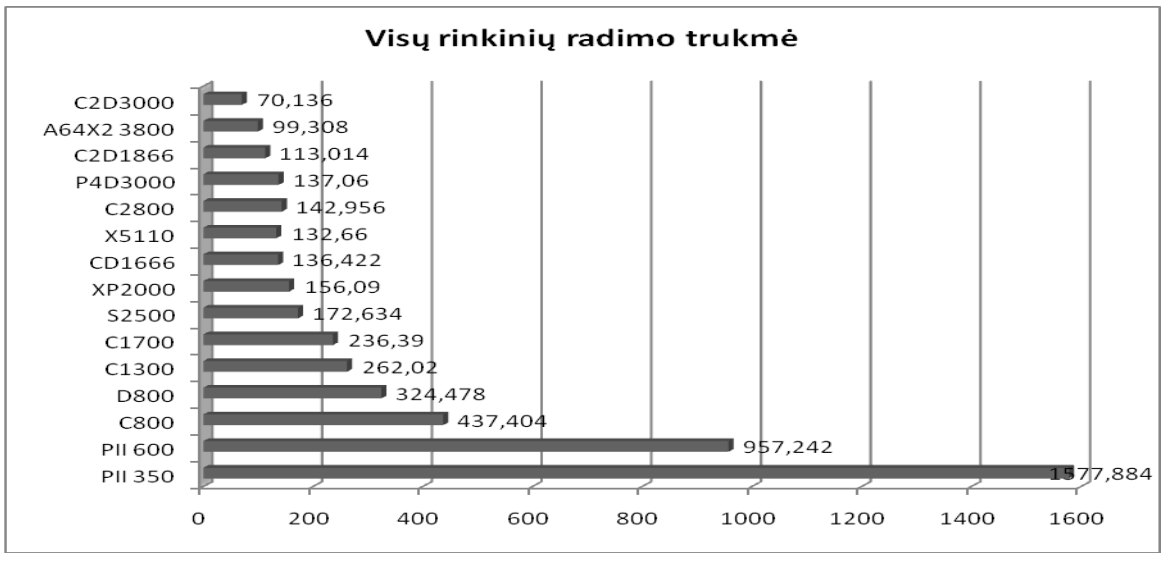
30 pav. c3540 Rastų K narių priklausomybė nuo generavimų skaičiaus

Kad rasti visus 2588 schemos c3540 unikalius K vidutiniškai tereikia atlikti 22000 generavimų. Tačiau ir toks ganėtinai mažasis generavimų kiekis nenulemia plataus K elementu išsibarstymo generavimų eigoje. Jau po 1% generavimų randamos 2482 K reikšmės, o tai jau 96% visų reikšmių. Po 10% generavimų jau rasta 99% ieškomu K.



31 pav. c3540 1.000.000 generavimų greitaveika

Šios schemos 1.000.000 generavimų greitaveika vidutinė (31 pav.) nuo 32s naudojant 3GHz Core2Duo 1 baranduolį iki 717s naudojant PentiumII 350Mhz.

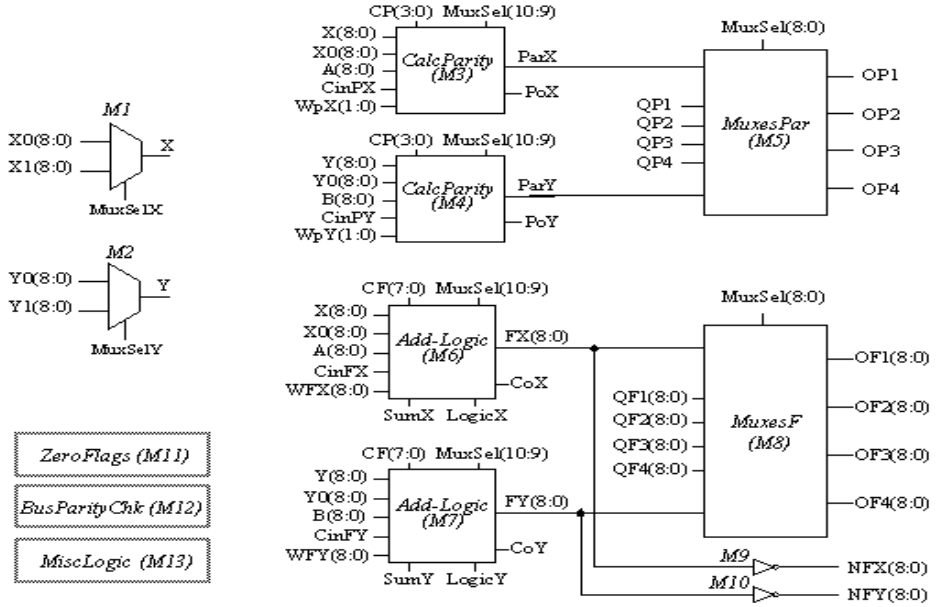


32 pav. c3540 22000 generavimų greitaveika

Ganėtinai dielė schemos generavimo sparta ir mažas reikiamų generavimų skaičius įtakoja sąliginai nedidelį visų unikalių K radimo laiką (32 pav.): nuo 70s iki 26min priklausomai nuo procesoriaus galingumo.

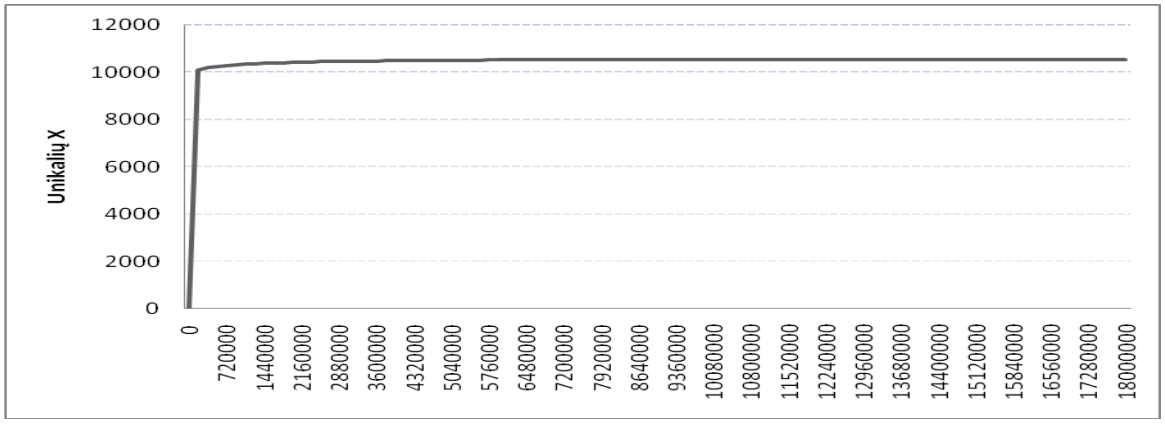
4.8. ISCAS-85 c5315

Statistika: 178 įėjimai; 123 išėjimai; 2406 ventiliai (1031 AND + 1035 NAND + 214 OR + 27 NOR); 581 inverteris; 313 buferių



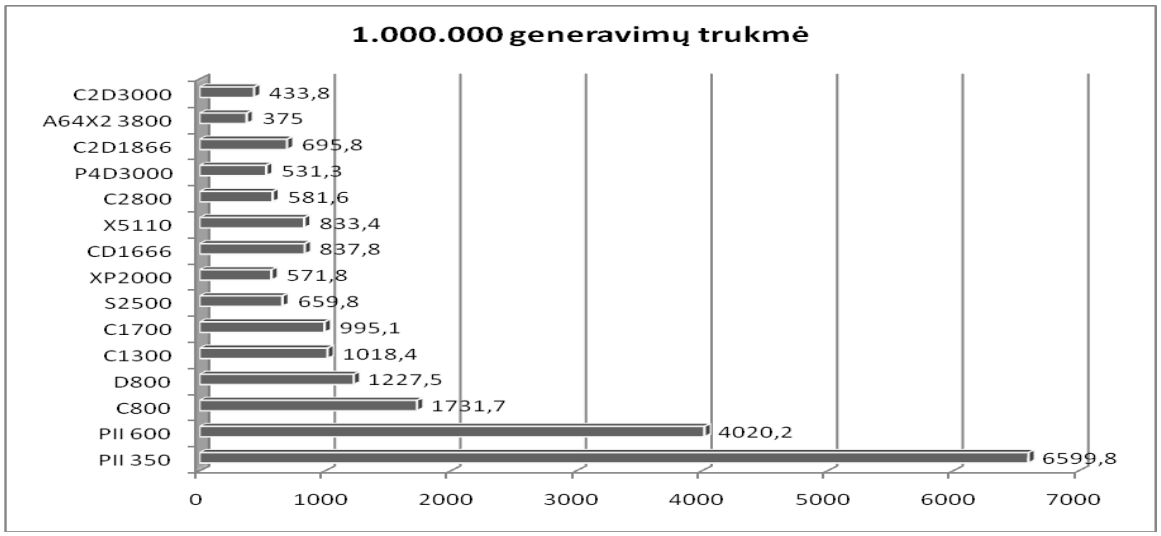
33 pav. c5315

ISCAS-85 c5315 yra 9-bitų ALU galintis atlikti aritmetine ir logines operacijas vienu metu dvejiems 9-bitų įėjimo žodžiams, o taip pat skaičiuoti rezultatų kontrolinę sumą. M5 ir M6 moduliai skaičiuoja aritmetines ar logines operacijas priklausomai nuo valdymo įėjimų CF[7:0]. M5 modulis sudarytas ir multiplekserių kurie yra tarpininkai tarp M6 ir M7 bei keturių įėjimų ir išėjimų. Išėjimų grandinės OF1 ir OF2 gali būti priskirtos loginiam nuliui priklausomai nuo MuxSel[8]. M3 ir M4modualiai skaičiuoja kontrolines sumas rezultatui CP=CF[7:4].



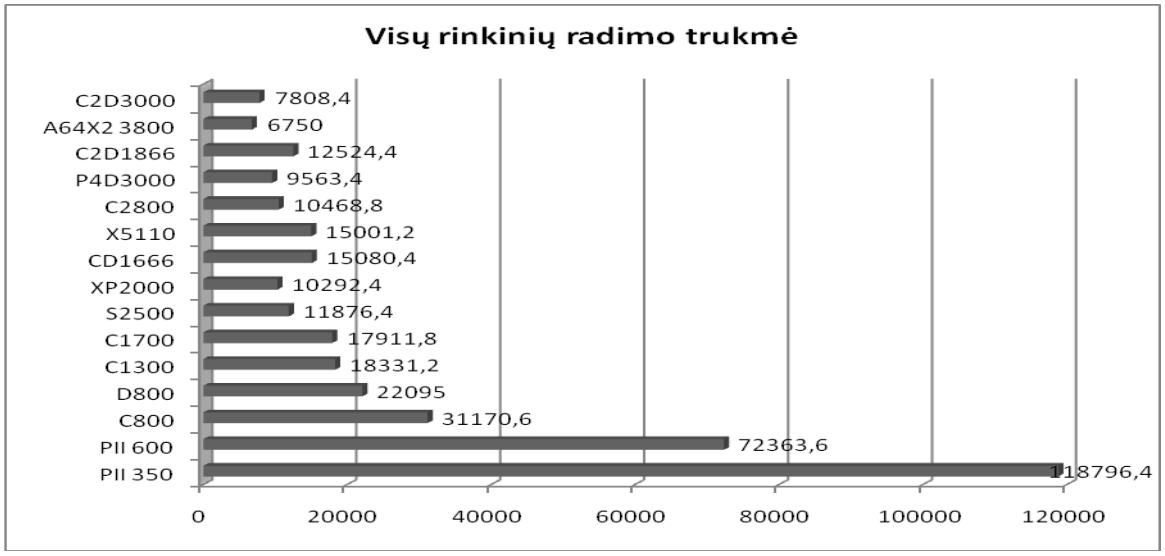
34 pav. c5315 Rastų K narių priklausomybė nuo generavimų skaičiaus

c5315 neskaitant c7552 reikalauja pačio didžiausio generavimo kiekio, net 18.000.000. Nors ir ieškomų K kiekis ir vienas iš didžiausių (10540), tačiau didelis generavimų kiekis nulemia daugumos unikalių K radimą jau generavimo pradžioje (34pav). Jau po 1% generavimų randama 96% ieškomų K, o jau po 10% randama net 99% ieškomų unikalių K.



35 pav. c5315 1.000.000 generavimų greitaveika

c5315 viena iš lėšiausiai generuojamų testuotų shemų. Jos milijonas generavimų užtrunka nuo 375s iki 6599s (35 pav.).

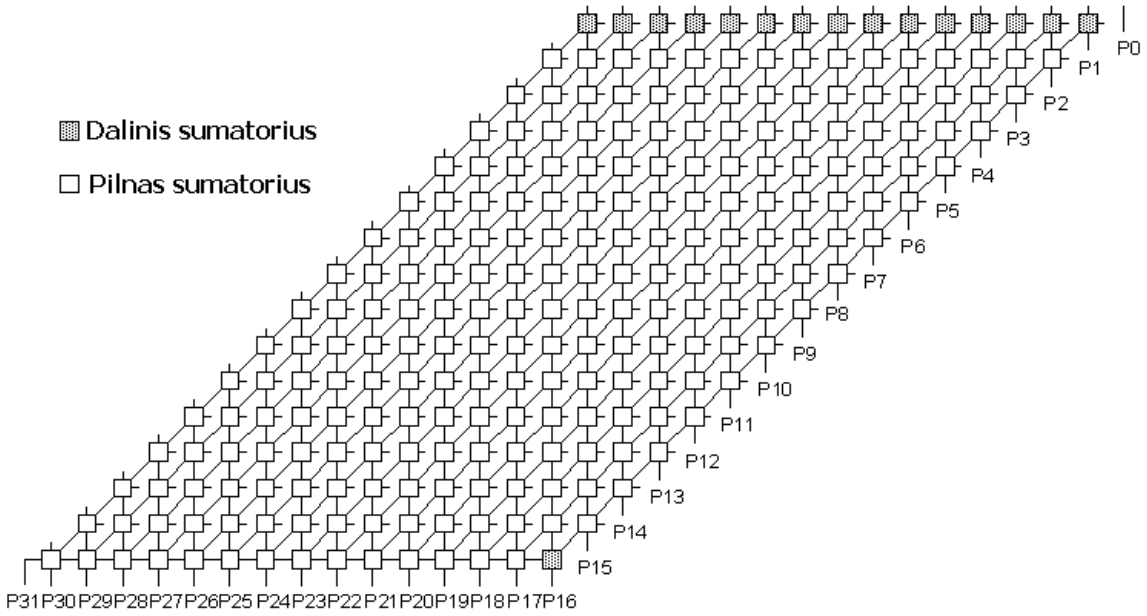


36 pav. c5315 18.000.000 generavimų greitaveika

Kadangi ši schema viena iš lėčiausiai skaičiuojamų, o generavimų skaičius vienas didžiausių tai visu ieškomų unikalių K radimas užtrunka nuo 1,875h iki 33h (36 pav.) priklausomai nuo pasirinkto procesoriaus spartos.

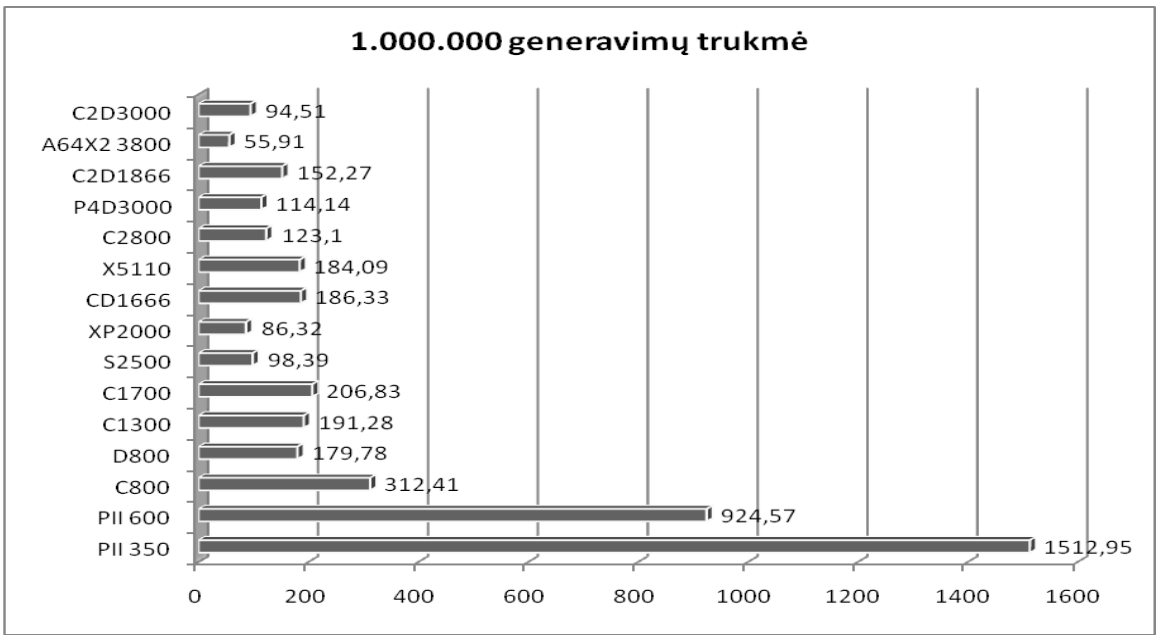
4.9. ISCAS-85 c6288

Statistika: 32 įėjimai; 32 išėjimai; 2406 ventiliai (256 AND + 32 NAND + 2128 NOR); 32 inverteris

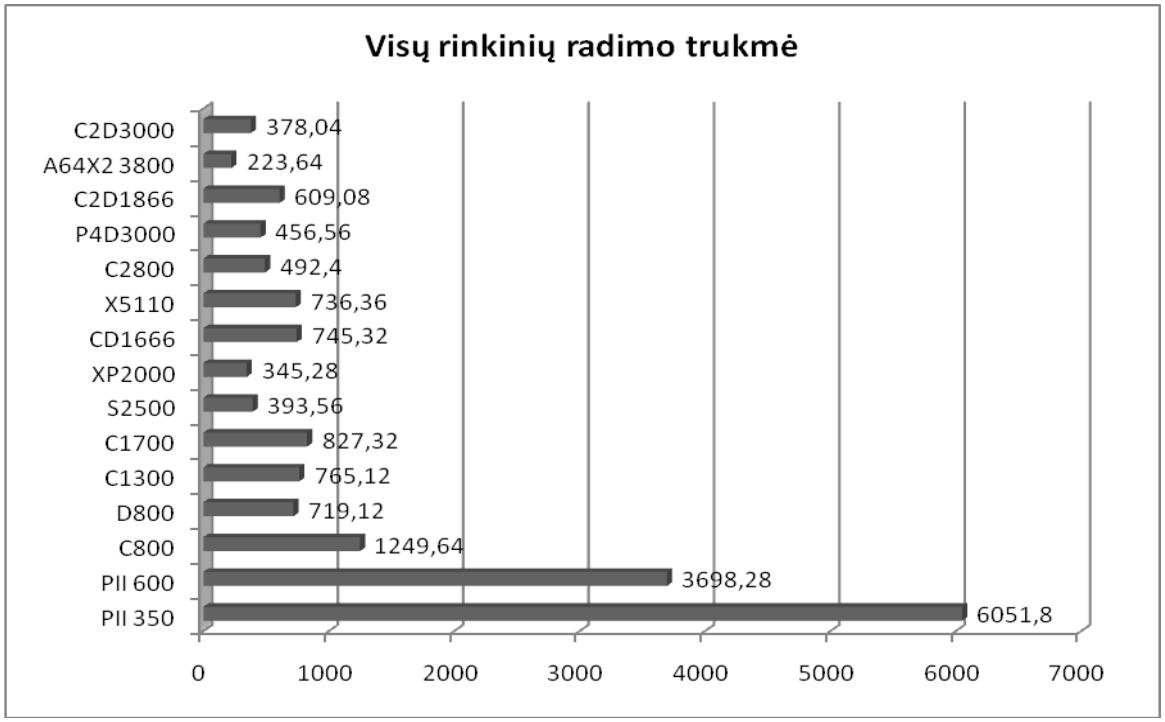


37 pav. c6288

ISCAS-85 c6288 yra multiplikacinė funkcija sudaryta iš 240 sumatorių apjungtų i 15x16 matricą.

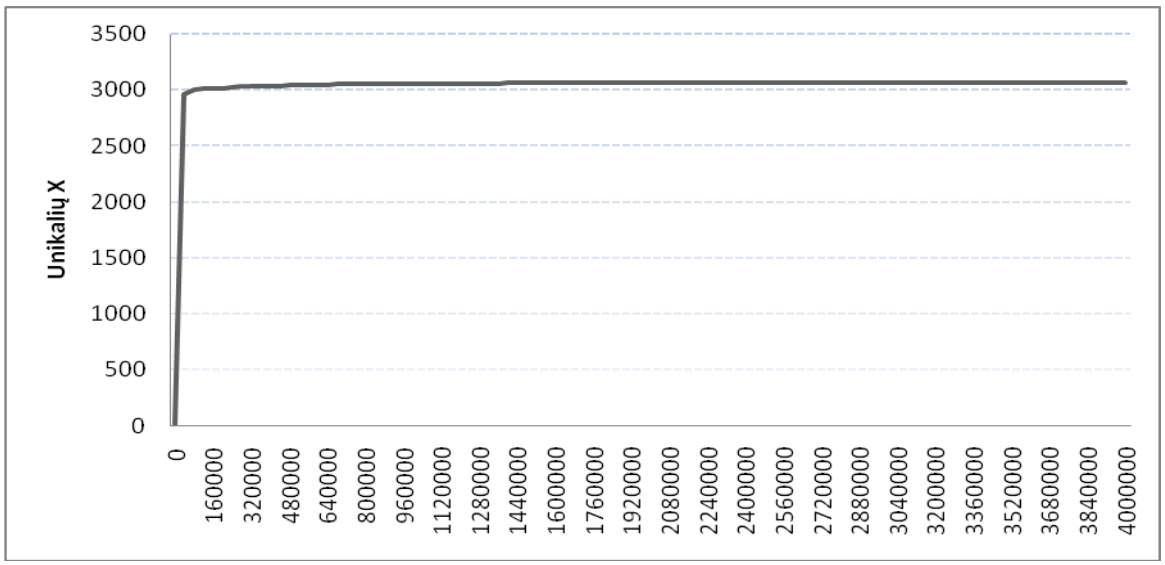


38 pav. c6288 1.000.000 generavimų greitaveika



39 pav. c6288 4000000 generavimų greitaveika

Nors c6288 ir turi mažiausiai iįėjimų tik 32, tačiau 240 vidinių sumatorių neleidžia pasiekti labai didelės skaičiavimo spartos (38 pav.). Kad rasti visus ieškomus 3068 unikalius K reikia atlikti 4.000.000 generavimų kurie užtrunka nuo 224s iki 101min priklausomai nuo to kokios spartos procesorių pasirinksime (39pav.).

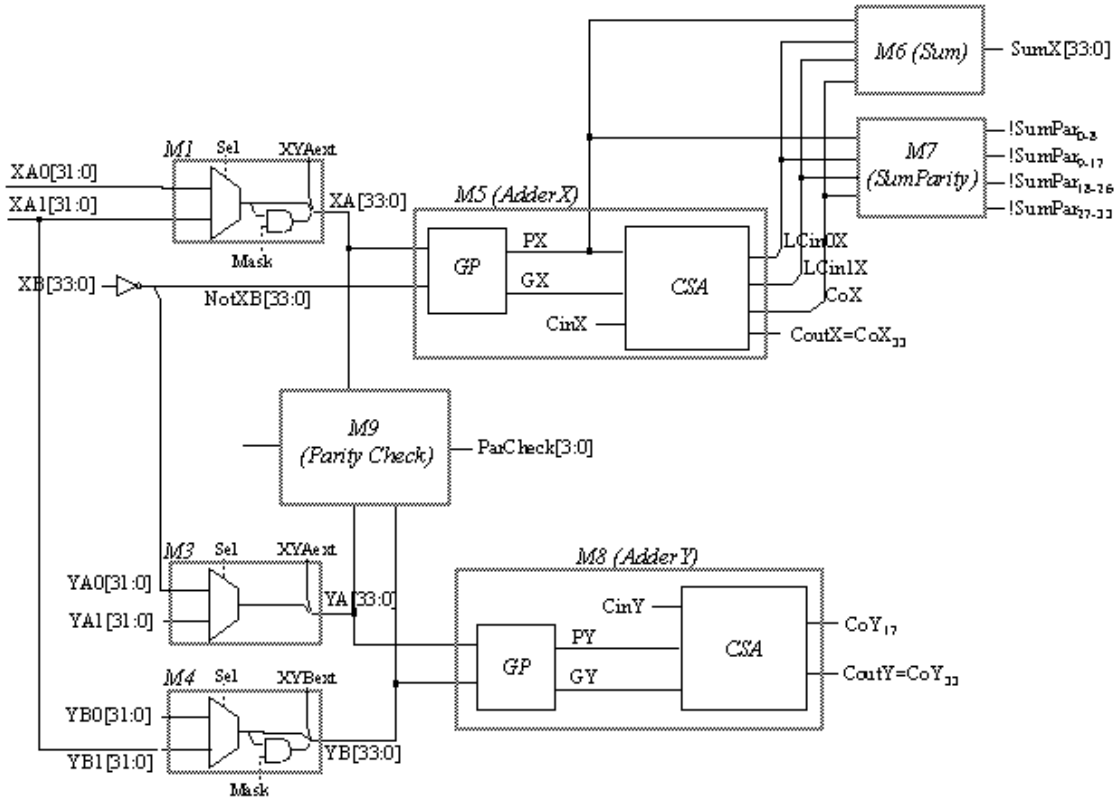


40 pav. c6288 Rastų K narių priklausomybė nuo generavimų skaičiaus

Sąlyginai nedidelis 3068 ieškomų unikalių K skaičius ir ganėtinai didelis keturių milijonų generavimų skaičius nulemia rastų unikalių K išsibarstymą generavimo pradžioje. Jau po 1% generavimų randama 97% ieškomų unikalių K (40 pav.)

4.10. ISCAS-85 c7552

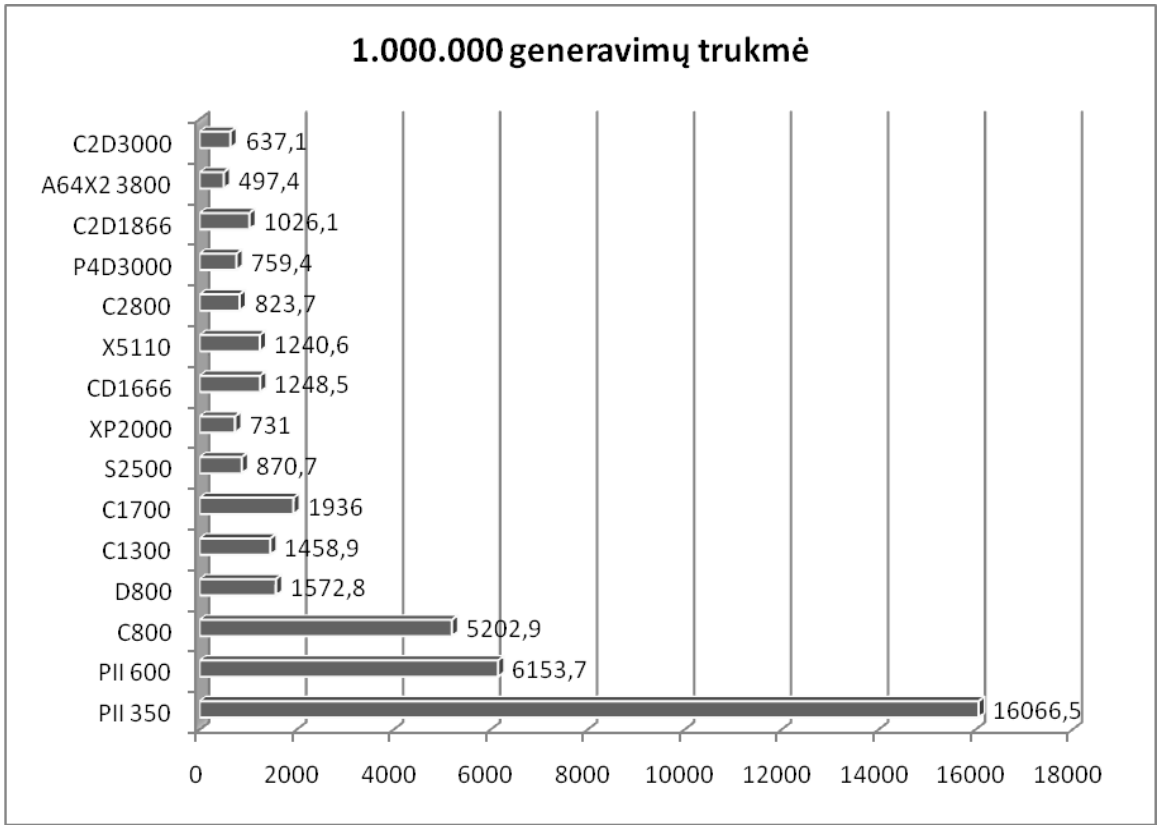
Statistika: 207 įėjimai; 107 išėjimai; 3512 ventiliai (1310 AND + 1904 NAND + 244 OR + 54 NOR); 876 inverteris; 534 buferis



41 pav. c7552

ISCAS-85 c7552 yra 34-bitų sumatorius ir dydžių sulyginimo funkcijos su įėjimo lyginumo kontrole. Schema sudaryta iš 34-bitų sumatoriaus (M5), 34-bitų dydžių sulyginimo funkcijos (M8) naudojančios kitą 34-bitų sumatorių ir lyginumo tikrintojo (M9). Kiekviena iš XA, YA ir YB magistralių aptarnaujama 2:1 multiplekserių valdomų Sel įėjimo. Sumatoriai M5 ir M8 yra identiški.

Ši schema yra pati didžiausia iš visų testų schemų visais atžvilgiais. Ieškomų matricos X elementų K daugiausia: 12188, schema turi daugiausia ventilių: 3512 ir kas ne mažiau svarbu, jog schema turi net 206 įėjimus. Teoriškai tokiai schemai galima sugeneruoti 2^206*2^205 o tai jau 8,9903611755473809457460481892457e+124. Toki rinkinių kiekį patikrinti nėra labai paprasta. Schemos tikrinimo greitaveika taipogi pati lėčiausia: nuo 497s iki 16067s milijonui generavimų (pav. 42)



42 pav. c7552 1.000.000 generavimų greitaveika

Nepaisant didžiausio ventilių ir išėjimų skaičiaus procentaliai schemai c7552 rasta 93,5% ieškomų rinkinių, kai, tuo tarpu, c2670 užstrigo ties 87,5%. Maža to rasti likusius c7552 matricos X unikalius K sutrukdė tik laiko ir skaičiavimo resursų trūkumas. Kadangi kiekviena papildoma įėjimo koja keturis kartus padaugina galimų testinių rinkinių skaičių, naturalu, jog šiai schemai ir reikia daugiausia skaičiavimo laiko, nors tam buvo skirta daugiau nei 1000 valandų.

4.11. Rezultatų suvestinė

Algoritmo veikimas patikrintas 35 schemose, iš kurių 10 schemų kontrolinės (c432, c499, c880, c1355, c1908, c2670, c3540, c5315, c6288, c7552), t.y. joms žinomas maksimalus matricos X unikalių K narių skaičius. Aštuonioms schemoms iš dešimties pavyko rasti visus unikalius K (1 lentelė).

1 lentelė Dešimties kontrolinių schemų rezultatai

Schema	Ventilių skaičius	Įėjimai n	Išėjimai m	K_{max},	Rasta K	%
C432	160	36	7	540	540	100
C499	202	41	32	5184	5184	100
C880	383	60	26	1326	1326	100
C1355	546	41	32	5184	5184	100
C1908	880	33	25	3004	3004	100
C2670	1193	157	64	3320	2904	87,5
C3540	1669	50	22	2588	2588	100
C5315	2307	178	123	10540	10540	100
C6288	2406	32	32	3068	3068	100
C7552	3512	206	107	12188	11401	93,5

Dvi schemos ištestuotos dalinai, c2670 rasta 2904 iš 3320 galimų, o c7552 rasta 11401 iš galimų 12188 unikalių matricos X elementų. Didelė tikimybė jog c2670 schemai nerastos visos reikšmės dėl algoritmo ar jo nagrinėto schemos duomenų failo defekto. Skaičiuojant c7552 pasirinktu algoritmu reikia daugiau skaičiavimo laiko arba daugiau skaičiavimo galios, nes generuojant rinkinius vis dar randama unikalių rinkinių.

2 lentelė Kontrolinių schemų visų unikalių K radimo greitaveikos rezultatai

Schema	Ventilių skaičius	Įėjimai n	Išėjimai m	Generavimų kiekis	MAKS laikas	MIN laikas
C432	160	36	7	7000	0,605549 s	0,047145 s
C499	202	41	32	30000	4,43754 s	0,36624 s
C880	383	60	26	400000	99,6432 s	8,2984 s
C1355	546	41	32	35000	11,26923 s	0,74816 s
C1908	880	33	25	13000000	37,4838 min	209,313 s
C3540	1669	50	22	22000	26,2981 min	70,136 s
C5315	2307	178	123	18000000	32,999 h	1,875 h
C6288	2406	32	32	4000000	100,864 min	223,64 s

Algortimo greitaveika kontorolinės schemose, kurių iėjimų skaičius nesiekia 100 iėjimų pakankamai didelė (2 lentelė). Naudojant sparčiausią testuose naudotą procesorių trys schemos (c432, c499, c1355) apdorojamos greičiau nei per 1 sekundę, kitoms keturioms (c880, c1908, c3540, c6288) užtenka mžaiau nei po 4 mintues, ir tik c5315 reikia beveik dviejų valandu skaičiavimo laiko, tačiau žinoma jog c5315 turi 178 įėjimus. Naudojant pati silpniausia testuose procesorių PentiumII 350Mhz galima rasti reikiamus rinkinius. Jo pagalba c432 galima apdorti greiciau nei per sekundę, o daugiausiai resursų reikalaujanti schema c5315 bus apdorota per šiekt tiek mažiau nei 33 valandas. Kitų procesorių skaičiavimo laikai pateikti prie schemų aprašymų.

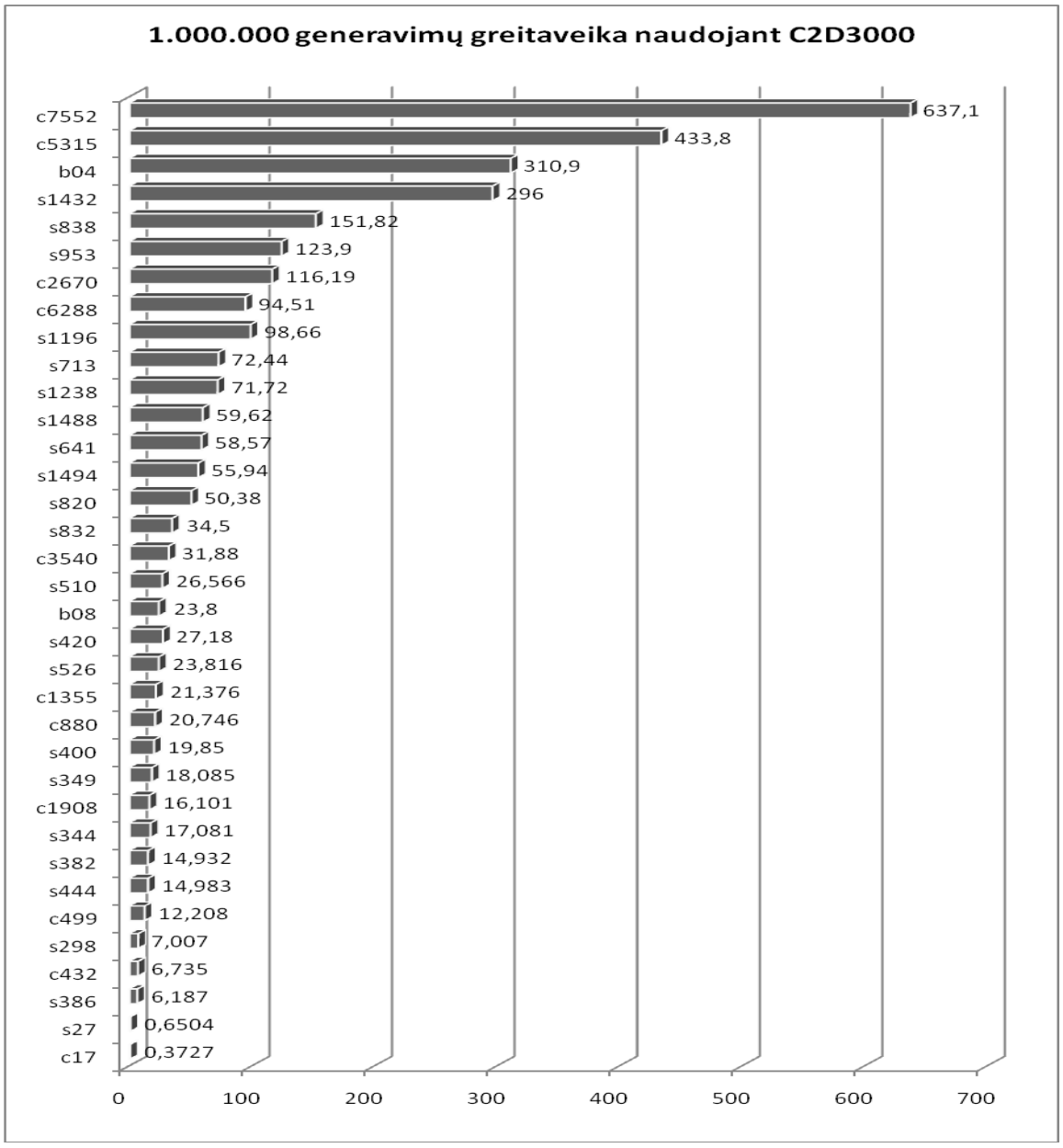
Be 10 kontrolinių schemų, algoritmas testuotas papildomose 25 schemose.

3 lentelė Papildomų testuojamų schemų rezultatai

Schema	Ventilių skaičius	Įėjimai n	Išėjimai m	Rasta K	Galimas K 100%	Generavimų kiekis	Min. Laikas	Maks. laikas
s27	8	7	4	45	T	400	0,000261	0,003667
s298	61	17	20	250	T	4000	0,028028	0,308048
s344	101	24	26	372	T	4000	0,068324	0,749068
s349	104	24	26	372	T	4000	0,07234	0,770744
s382	99	24	27	516	T	500000	3,0935	35,8905
s386	118	13	13	318	T	1300000	8,0431	93,3153
s400	106	24	27	548	T	60000	1,191	12,49554
s420	122	35	18	434	T	600000	16,308	184,332
s444	119	24	27	548	T	60000	0,89898	10,52598
s510	179	25	13	326	T	8000	0,212528	2,120656
s526	141	24	27	552	T	50000	1,1908	12,052
s641	107	54	43	1152	T	15000000	14,64min	2,66h
s713	139	54	42	2162	N	-		
s820	256	23	12	532	T	1300000	65,494	655,07
s832	262	23	12	532	T	1300000	44,85	436,514
s838	241	67	34	873	N	-		
s953	311	45	52	963	N	-		
s1196	388	32	32	1202	N	-		
s1238	428	32	32	1429	N	-		
s1423	490	91	79	14259	N	-		
s1488	550	14	25	836	T	300000	17,886	181,323
s1494	558	14	25	836	T	300000	16,782	170,124
b04	632	77	74	2799	N	-		
b08	157	30	25	448	T	130000	3,094	36,0061
c17	6	5	2	18	T	200	7,45E-05	0,001056

Trečioje lentelėje pateikta likusių 25 shemų testavimo rezultatai. Penktame stulpelyje pateikta kiek matricos X unikalių reikšmių rasta testavimo metu, o ar tai galimas schemos maksimusmas nurodyta šeštame laukelyje. Jei numanomas

maksimumas tai septintame laukelyje nurodyta kiek vidutiniškai reikia atlikti generavimų kad pasiekti numanomą maksimumą, o aštuntame ir devintame laukeliuose nurodomas minimalus ir maksimalus laikas reikiamiems generavimams ivykdyti. Iš 25 schemų didelė tikimybė, jog 18 rastos visos unikalios K reikšmės.



43 pav. C2D3000 visų schemų 1.000.000 generavimo laikas

Schemų tikrinimo greitaveika priklauso nuo ventilių skaičiaus ir iėjimų bei išėjimų skaičiaus (43 pav.) Kuo daugiau ventilių, tuo ilgiaužtrunka schemos reakcijos į pokytį tikrinimas. Kuo įejimų skaičius didesnis, tuo daugiau testinių rinkinių reikia genreuoti, o esant dideliamiėjimų ir išėjimų skaičiui ilgėja pokyčių tikrinimas ir rezultatų tikgrinimas bei saugojimas X matricoje (2nx2m).

4. Išvados

1. Sukurtas ir realizuotas algortimas gebantis rasti rinkinius tinkamus schemų testavimui.
2. Aštuoniose kontrolinėse schemose (c432, c499, c880, c1355, c1908, c3540, c5315, c6288) testavimo metu rasti visi ieškomi testiniai rinkianiai.
3. c7552 schemos testinių rinkinių paieškai nepakako skaičiavimo spartos, skaičiavimo laiko.
4. c2670 schemoje nerasti visi testiniai rinkiniai dėl tiriamo schemos failo, ar algoritmo klaidų.
5. Nedidelių schemų skaičiavimai nereikalauja spartaus procesoriaus.
6. Per pirmus 10% visų reikiamų generavimų randama iki 99% ieškomų pokyčių.
7. Schemų analizės greitaveiką smarkiai įtakoja įėjimų ir išėjimų skaičius.

5. Literatūra:

[1] G. L. Smith, “Model for Delay Faults Based Upon Paths,” IEEE Int’l Test Conf., Philadelphia, PA, Oct. 1985, pp. 342-349.

[2] C. J. Lin and S. M. Reddy, “On Delay Fault Testing in Logic Circuits,” IEEE Trans. on Computer-Aided Design, vol. 6, no. 9, Sept. 1987, pp. 694-701.

[3] J. A. Bell, “Timing Analysis of Logic-Level Digital Circuits Using Uncertainty Intervals,” M. S. Thesis, Department of Computer Science, Texas A&M University, 1996.

[4] W. N. Li, S. M. Reddy and S. K. Sahni, “On Path Selection in Combinational Logic Circuits,” IEEE Trans. On Computer-Aided Design, vol. 8, no. 1, Jan. 1989, pp. 56-63.

[5] A. K. Majhi, V. D. Agrawal, J. Jacob and L. M. Patnaik, “Line Coverage of Path Delay Faults,” IEEE Trans. on VLSI Systems, vol. 8, no. 5, Oct. 2000, pp. 610-613.

[6] A. Murakami, S. Kajihara, T. Sasao, R. Pomeranz and S. M. Reddy, “Selection of Potentially Testable Path Delay Faults for Test Generation,” IEEE Int’l Test Conf., Atlantic City, NJ, Oct. 2000, pp. 376-384.

[7] Y. Shao, S. M. Reddy, I. Pomeranz and S. Kajihara, “On Selecting Testable Paths in Scan Designs,” IEEE European Test Workshop, Corfu, Greece, May 2002, pp. 53-58.

[8] M. Sharma and J. H. Patel, “Finding a Small Set of Longest Testable Paths that Cover Every Gate,” IEEE Int’l Test Conf., Baltimore, MD, Oct. 2002, pp. 974-982.

[9] V. Iyengar, B. K. Rosen and I. Spillinger, “Delay Test Generation 1 – Concepts and Coverage Metrics,” IEEE Int’l Test Conf., Washington, DC, Sept. 1988, pp. 857-866.

[10] J. L. Carter, V. S. Iyengar and B. K. Rosen, “Efficient Test Coverage Determination for Delay Faults,” IEEE Int’l Test Conf., Washington, DC, Sept. 1987, pp. 418-427.

[11] G. M. Luong and D. M. H. Walker, “Test Generation for Global Delay Faults,” IEEE Int’l Test Conf., Washington, DC, Oct. 1996, pp. 433-442.

[12] T. W. Williams, B. Underwood and M. R. Mercer, “The Interdependence Between Delay-Optimization of Synthesized Networks and Testing,” ACM/IEEE Design Automation Conf., San Francisco, CA, June 1991, pp. 87-92.

[13] J. Benkoski, E. V. Meersch, L. J. M. Claesen and H. D. Man, “Timing Verification Using Statically Sensitizable Paths,” IEEE Trans. on Computer-Aided Design, vol. 9, no. 10, Oct. 1990, pp. 1073-1084.

[14] P. McGeer and R. K. Brayton, “Efficient Algorithms for Computing the Longest Viable Path in a Combinational Network,” ACM/IEEE Design Automation Conf., Las Vegas, NV, June 1989, pp. 561-567.

[15] H. Chang and J. A. Abraham, “VIPER: An Efficient Vigorously Sensitizable Path Extractor,” ACM/IEEE Design Automation Conf., Dallas, TX, June 1993, pp. 112-117.

[16] J. J. Liou. A. Krstic, Li-C. Wang and K. T. Cheng, “False- Path-Aware Statistical Timing Analysis and Efficient Path Selection for Delay Testing and Timing Validation,” ACM/IEEE Design Automation Conf., New Orleans, LA, June 2002, pp. 566-569.

[17] I. Pomeranz, S. M. Reddy and P. Uppaluri, “NEST: A Nonenumerative Test Generation Method for Path Delay Faults in Combinational Circuits,” IEEE Trans. On Computer-Aided Design, vol. 14, no. 12, Dec. 1995, pp. 1505-1515.

[18] K. Fuchs, F. Fink and M. H. Schulz, “DYNAMITE: An Efficient Automatic Test Pattern Generation System for Path Delay Faults,” IEEE Trans. on Computer-Aided Design, vol. 10, no. 10, Oct. 1991, pp. 1323-1355.

[19] K. Fuchs, M. Pabst and T. Rossel, “RESIST: A Recursive Test Pattern Generation Algorithm for Path Delay Faults Considering Various Test Classes,” IEEE Trans. On Computer-Aided Design, vol. 13, no. 12, Dec. 1994, pp. 1550-1562.

[20] R. Stewart and J. Benkoski, “Static Timing Analysis Using Interval Constraints,” IEEE Int’l Conf. on Computer-Aided Design, Santa Clara, CA, June 1991, pp. 308-311.

[21] An Efficient Algorithm for Finding the K Longest Testable Paths Through Each Gate in a Combinational Circuit Wangqi Qiu D. M. H. Walker Department of Computer Science Texas A&M University

[22] J.Savir and S. Patil, “Scan-Based Transition Test,” IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Volume: 12 , Issue: 8 , Aug. 1993

[23] M. L.Bushnell and V. D. Agrawal, “Essentials of Electronic Testing for Digital, Memory, and Mixed-Signal VLSI Circuits”, Kluwer Academic Publishers, Boston, 2000.

[24] Eduardas Bareiša, Vacius Jusas, Kęstutis Motiejūnas, Rimantas Šeinauskas „FUNCTIONAL DELAY TEST QUALITY ASSESSMENT ON HIGH LEVEL OF ABSTRACTION“

[25] Jiang Brandon Liu, Andreas Veneris, Sean Safarpour "Diagnosing Multiple Transition Faults in the Absence of Timing Information"

[26] Nandu Tendolkar, Dawit Belete, Bill Schwarz, Bob Podnar, Akshay Gupta, Steve Karako, Wu-Tung Cheng, Alex Babin, Kun-Han Tsai, Nagesh Tamarapalli, Greg Aldrich „Improving Transition Fault Test Pattern Quality through At-Speed Diagnosis“

Priedai

Schemų generavimo spartos priklausomybės nuo pasirinkto procesoriaus lentelė

4 lentelė Procesorių greitaveika schemose skaičiuojant 1.000.000 generavimų

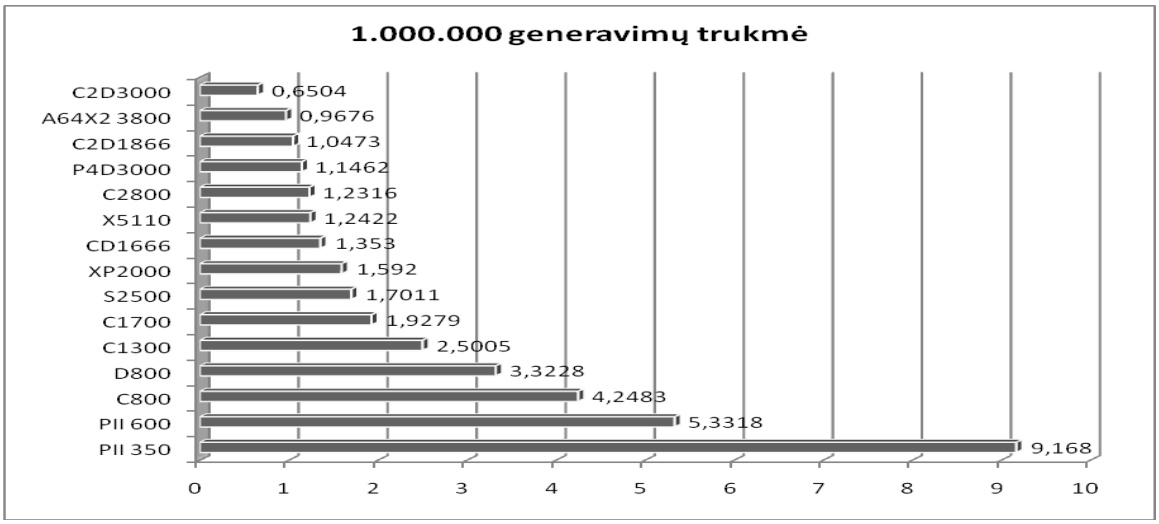
	PII 350	PII 600	C800	D800	C1300	C1700	S2500	XP2000
c17	5,2781	3,074	2,3583	1,935	1,4425	1,0833	1,0645	0,9315
s27	9,168	5,3318	4,2483	3,3228	2,5005	1,9279	1,7011	1,592
s386	71,781	41,741	32,035	26,86	19,314	16,687	15,092	12,921
c432	86,507	50,372	38,491	30,285	23,404	21,435	17,295	14,571
s298	77,012	44,783	34,569	27,402	20,56	17,278	16,591	13,186
c499	147,918	86,313	65,518	55,009	39,599	35,571	30,785	26,019
s444	175,433	102,096	78,449	65,567	47,453	39,613	35,89	31,546
s382	161,912	94,491	73,806	61,902	43,807	37,076	37,056	29,768
s344	187,267	108,968	83,677	66,837	50,448	50,924	41,357	32,167
c1908	173,002	100,523	76,151	62,428	45,234	42,563	36,824	30,009
s349	192,686	112,209	86,291	70,271	52,527	56,079	38,256	33,81
s400	208,259	121,159	93,415	76,787	56,467	56,842	46,902	36,953
c880	249,108	145,068	110,42	91,618	66,672	65,378	53,386	44,044
c1355	321,978	188,714	122,982	91,716	73,696	72,32	50,572	43,752
s526	241,04	140,244	109,472	96,514	65,236	71,098	56,31	46,416
s420	307,22	178,72	137,17	112,98	82,88	86,64	65,85	54,37
b08	276,97	154,41	118,54	94,42	70,95	75,39	57,54	45,36
s510	265,082	154,162	118,666	97,408	72,524	82,39	61,892	46,852
c3540	717,22	435,11	198,82	147,49	119,1	107,45	78,47	70,95
s832	335,78	195,72	149,73	141,91	89,27	120,25	82	68,18
s820	503,9	291,06	222,86	228,56	132,59	147,06	123,17	110
s1494	567,08	323,75	246,98	240,21	150,6	139,63	125,71	115,88
s641	640,15	372,69	291,06	229,76	171,79	175,29	138,26	110,46
s1488	604,41	345,55	264,31	255,16	159,93	149,4	134,15	122,87
s1238	774,14	424,7	323,87	294,31	193,76	329,83	168,13	141,5
s713	754,7	439,51	338,89	294,53	204,12	218,8	162,27	141,62
s1196	1032,04	574,03	470,85	379,48	265,86	398,73	223,09	182,29
c6288	1512,95	924,57	312,41	179,78	191,28	206,83	98,39	86,32
c2670	1577,07	921,34	644,21	532,37	385,53	413,51	292,28	248,02
s953	1364,82	763,13	592,71	479,44	356,41	465,13	286,44	230,73
s838	1628,2	946,72	726,41	586,38	443,38	565,25	353,47	282,23
s1432	3187,5	1821,2	1385,2	1114,2	836,9	917,2	670,9	530,5
b04	3614,9	2090,5	1699,6	1286	949,5	941,3	735,3	613,3
c5315	6599,8	4020,2	1731,7	1227,5	1018,4	995,1	659,8	571,8
c7552	16066,5	6153,7	5202,9	1572,8	1458,9	1936	870,7	731

5 lentelė Procesorių greitaveika schemose skaičiuojant 1.000.000 generavimų

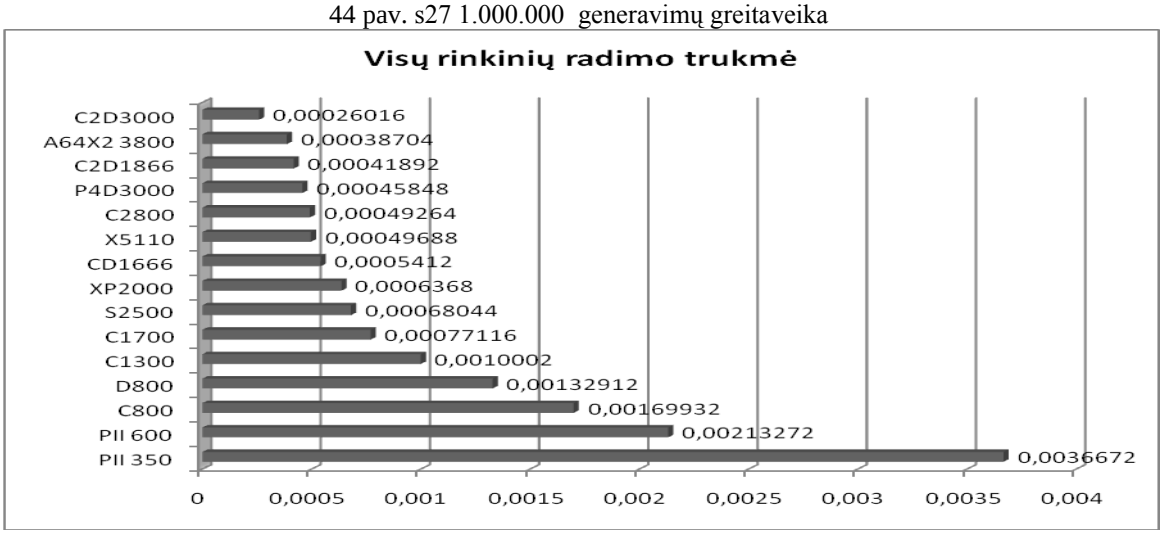
	CD1666	X5110	C2800	P4D3000	C2D1866	A64X2 3800	C2D3000
c17	0,7838	0,702	0,7357	0,6887	0,6006	0,6131	0,3727
s27	1,353	1,2422	1,2316	1,1462	1,0473	0,9676	0,6504
s386	11,981	11,678	10,119	9,799	9,966	8,722	6,187
c432	13,567	12,84	12,791	11,932	10,84	9,96	6,735
s298	13,303	13,238	10,802	10,161	11,291	9,562	7,007
c499	25,726	23,303	22,173	20,777	19,671	17,729	12,208
s444	30,906	28,577	23,025	22,07	24,129	20,637	14,983
s382	28,876	28,105	22,298	20,753	24,039	21,398	14,932
s344	33,404	32,17	27,741	26,547	27,498	23,858	17,081
c1908	30,472	30,564	27,571	26,167	25,931	21,258	16,101
s349	35,265	34.109	30.737	29.059	29,135	24,846	18,085
s400	37,784	37,35	33,097	31,417	31,97	27,116	19,85
c880	43,426	39,094	37,376	35,428	33,406	30,646	20,746
c1355	45,15	40,39	42,396	39,708	34,476	28,712	21,376
s526	45,042	45,144	39,776	37,968	38,346	32,446	23,816
s420	54,6	51,24	52,05	50,44	43,77	37,98	27,18
b08	47,47	45,07	42,1	40,19	38,33	32,87	23,8
s510	49,258	50,154	47,24	45,082	42,786	35,624	26,566
c3540	62,01	60,3	64,98	62,3	51,37	45,14	31,88
s832	64,27	65,21	65,6	64,63	55,62	47,32	34,5
s820	92,15	94,94	85,42	81,76	81,19	70,76	50,38
s1494	102,62	105,34	92,39	87,12	90,02	72,4	55,94
s641	113,45	110,58	99,73	94,93	94,49	79,78	58,57
s1488	109,09	112,86	100,84	94,95	96,02	77,14	59,62
s1238	135,2	135,3	185,09	178,06	115,41	96,83	71,72
s713	138,81	136,41	117,14	110,42	116,79	93,57	72,44
s1196	186,1	185,98	232,08	230,05	158,96	128,83	98,66
c6288	186,33	184,09	123,1	114,14	152,27	55,91	94,51
c2670	228,27	220,54	231,25	214,04	187,11	167,93	116,19
s953	240,56	232,26	260,03	256,11	199,5	165,16	123,9
s838	299,29	286,49	319,96	314,24	244,62	203,98	151,82
s1432	546,4	560,4	560,2	536,4	477,5	385,9	296
b04	631,7	586,7	572,4	539,5	501,6	423,9	310,9
c5315	837,8	833,4	581,6	531,3	695,8	375	433,8
c7552	1248,5	1240,6	823,7	759,4	1026,1	497,4	637,1

ISCAS-89 s27

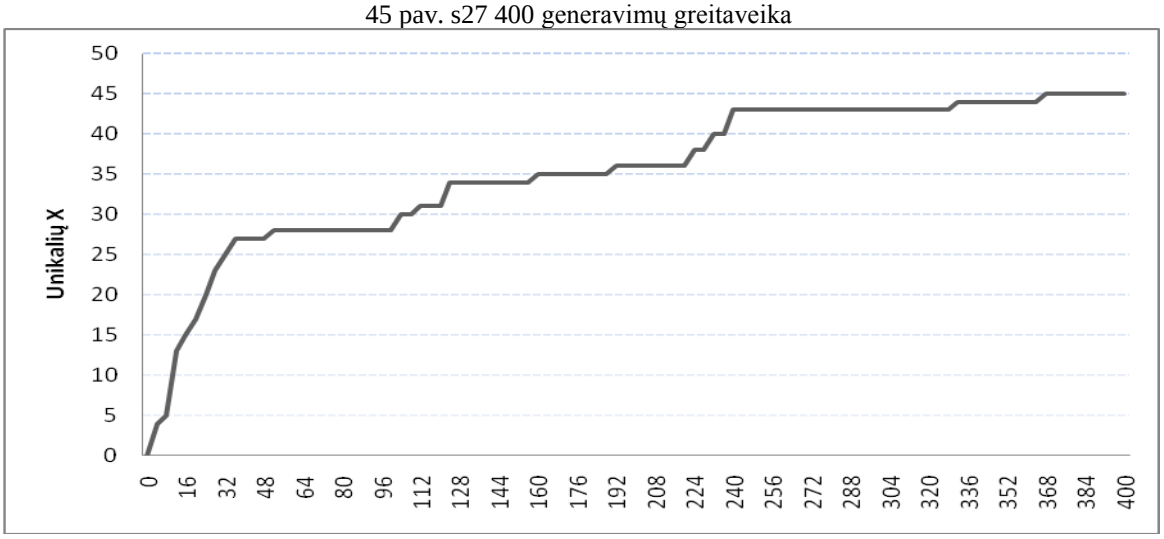
Statistika: 4 įėjimai; 1 išėjimas; 8 ventiliai (1 AND + 1 NAND + 2 OR + 4 NOR); 2 inverteriai; 3 trigeriai



44 pav. s27 1.000.000 generavimų greitaveika



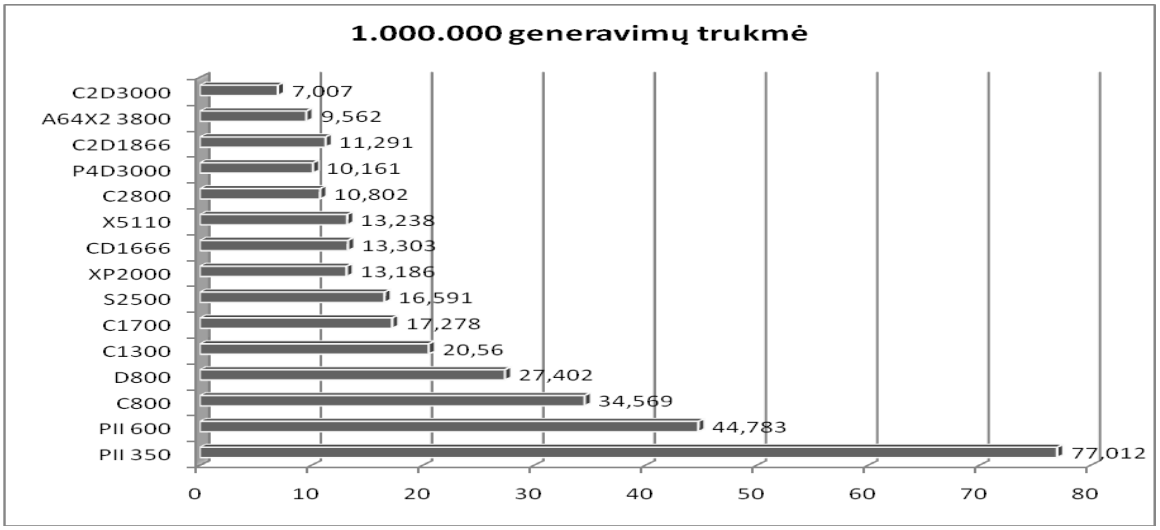
45 pav. s27 400 generavimų greitaveika



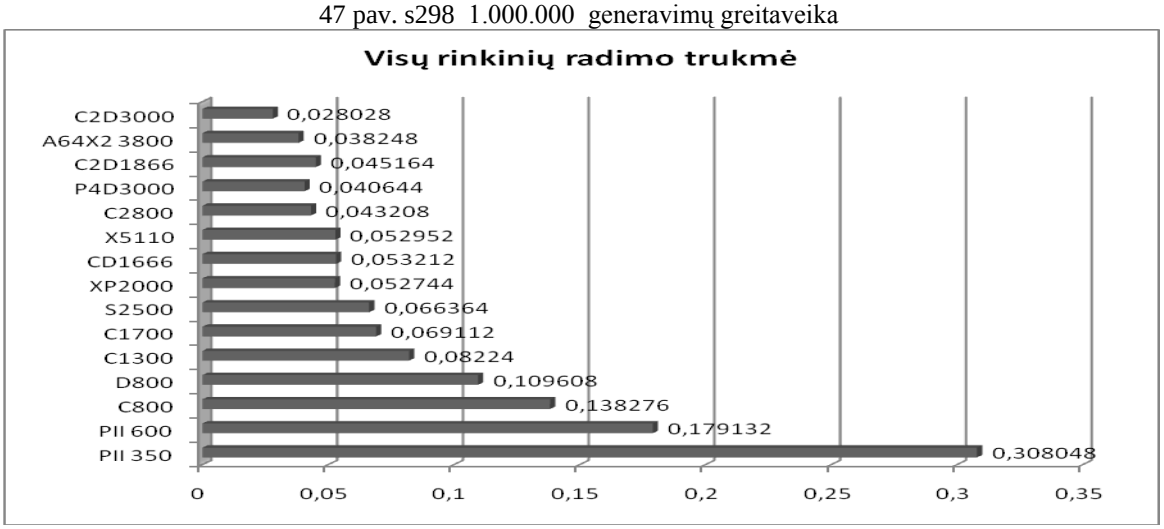
46 pav. s27 rastų K narių priklausomybė nuo generavimų skaičiaus

ISCAS-89 s298

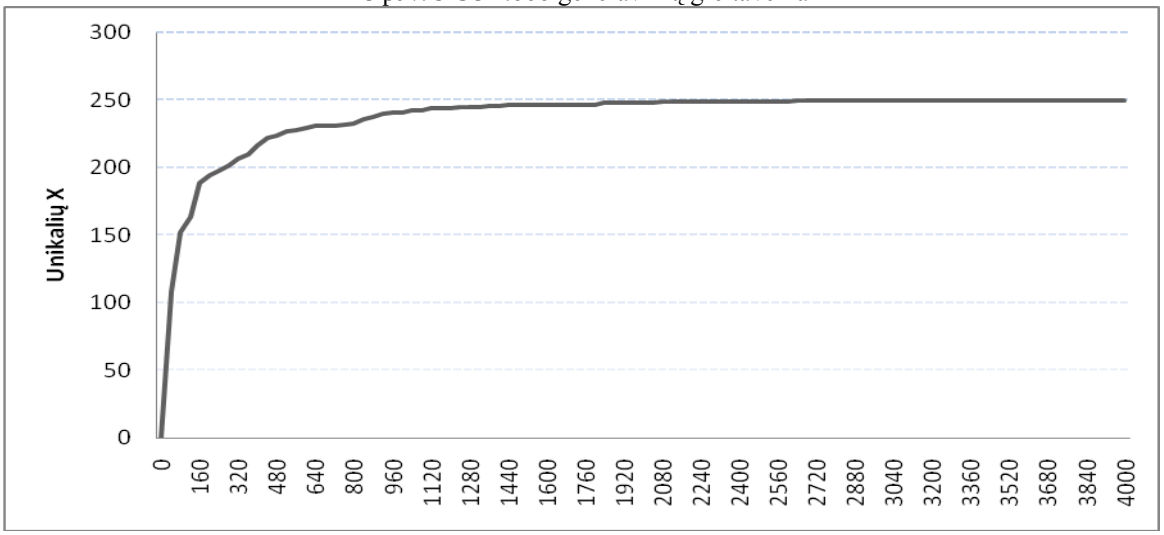
Statistika: 3 įėjimai; 6 išėjimai; 75 ventiliai (31 AND + 9 NAND + 16 OR + 19 NOR); 44 inverteriai; 14 trigerių



47 pav. s298 1.000.000 generavimų greitaveika



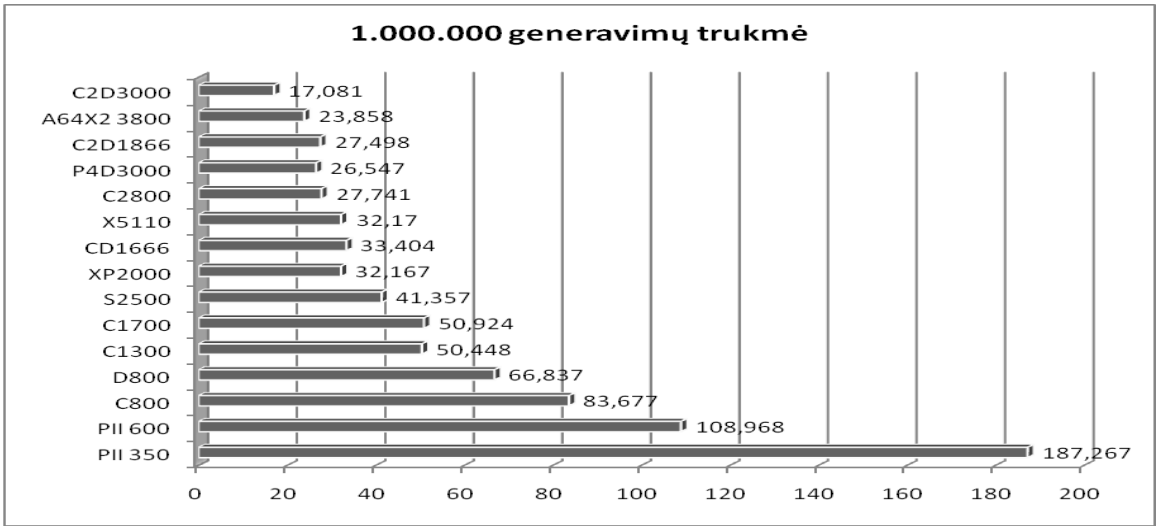
48 pav. s298 4.000 generavimų greitaveika



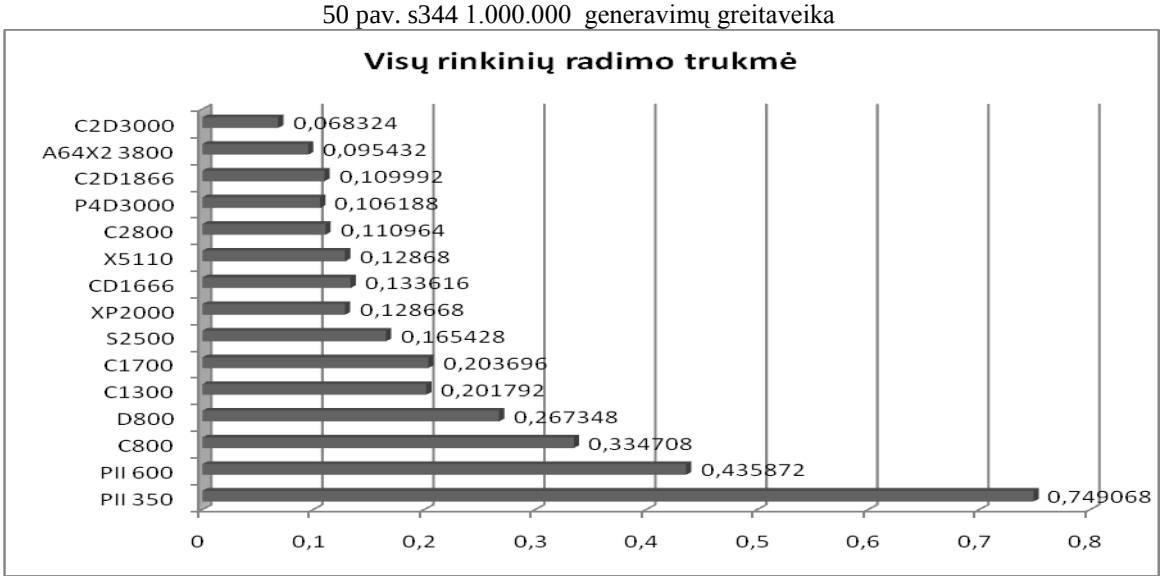
49 pav. s298 rastų K narių priklausomybė nuo generavimų skaičiaus

ISCAS-89 s344

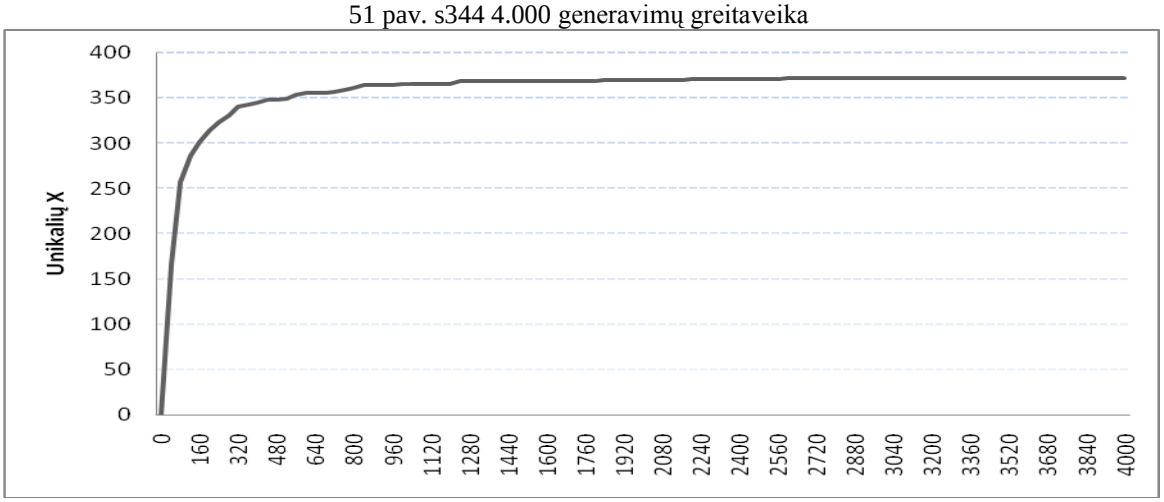
Statistika: 9 įėjimai; 15 išėjimų; 59 ventiliai (44 AND + 18 NAND + 9 OR + 30 NOR); 59 inverteriai; 15 trigerių



50 pav. s344 1.000.000 generavimų greitaveika



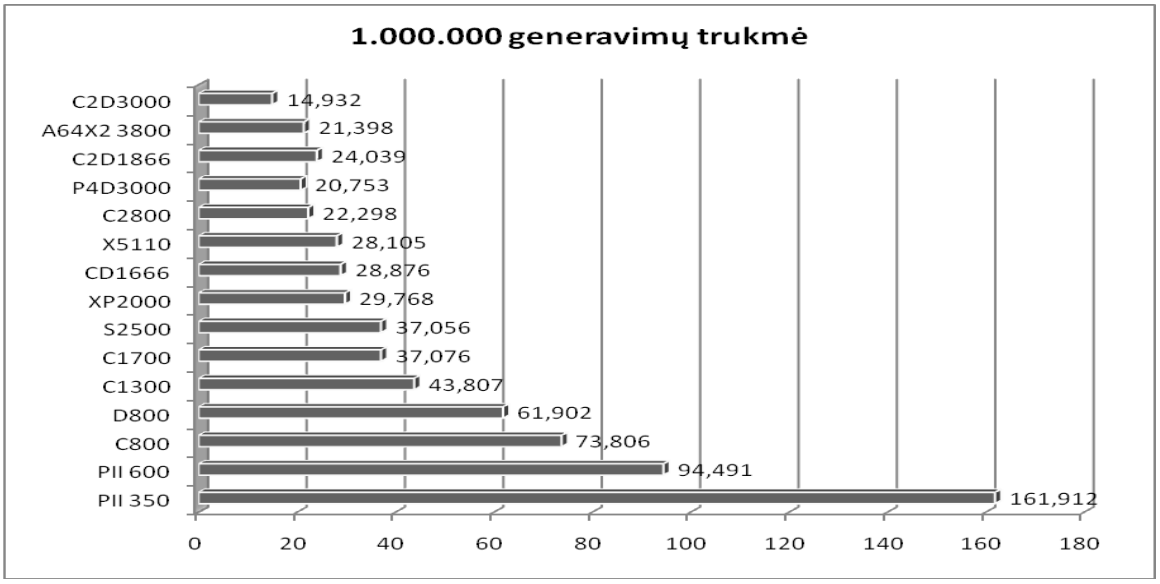
51 pav. s344 4.000 generavimų greitaveika



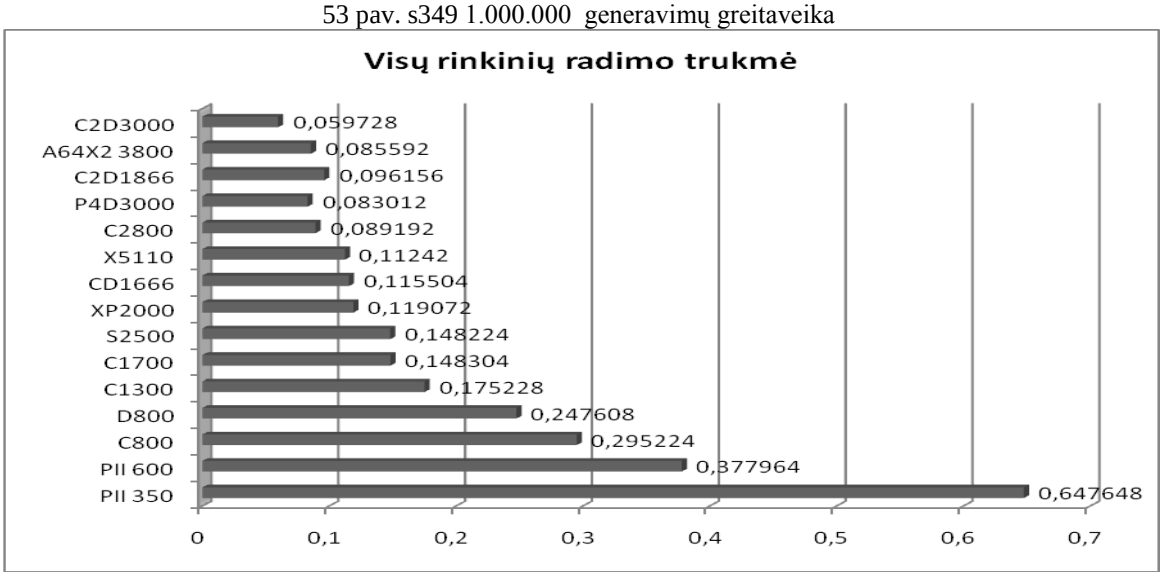
52 pav. s344 rastų K narių priklausomybė nuo generavimų skaičiaus

ISCAS-89 s349

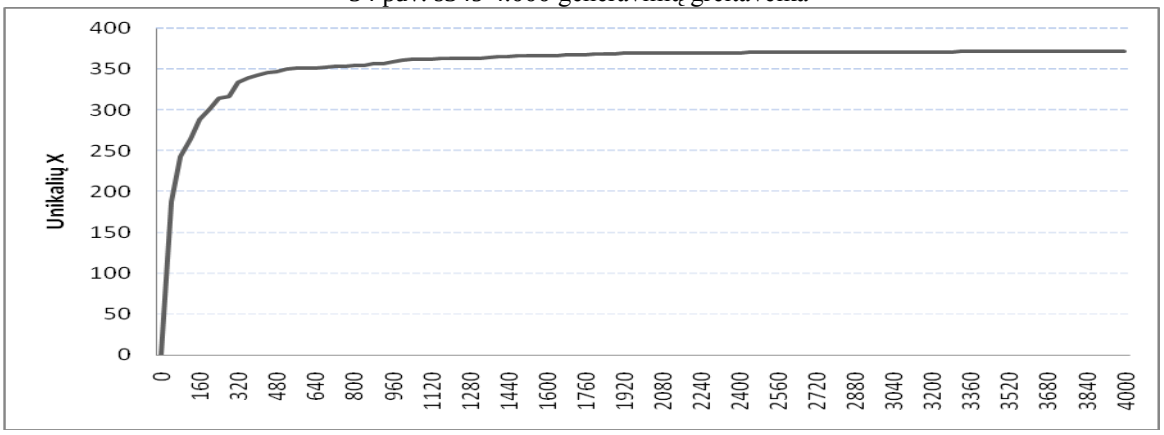
Statistika: 9 įėjimai; 11 išėjimų; 104 ventiliai (44 AND + 19 NAND + 10 OR + 31 NOR); 57 inverteriai; 15 trigerių



53 pav. s349 1.000.000 generavimų greitaveika



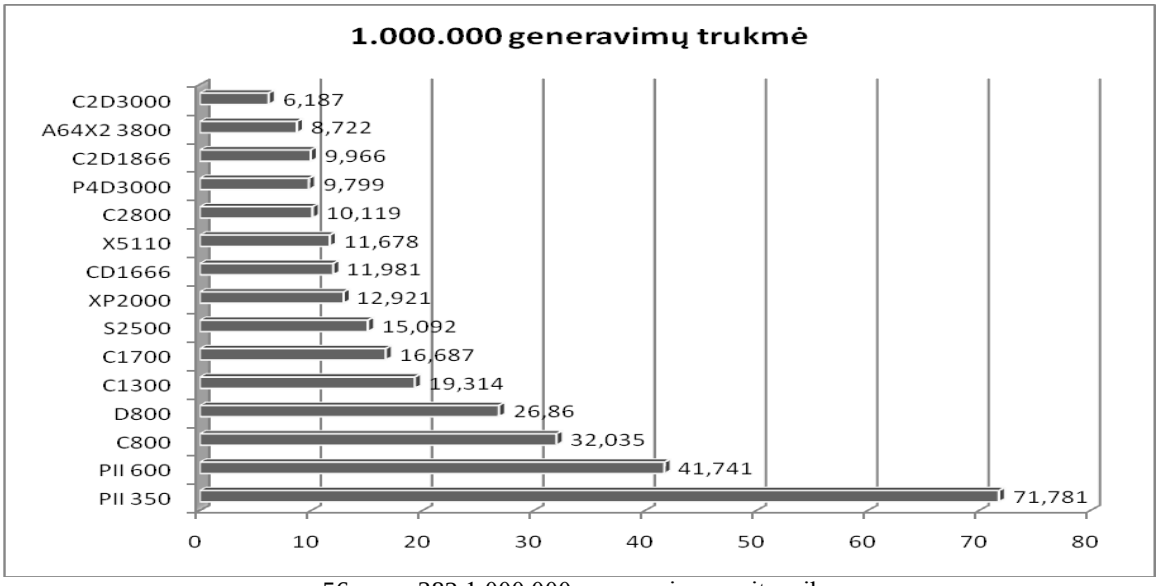
54 pav. s349 4.000 generavimų greitaveika



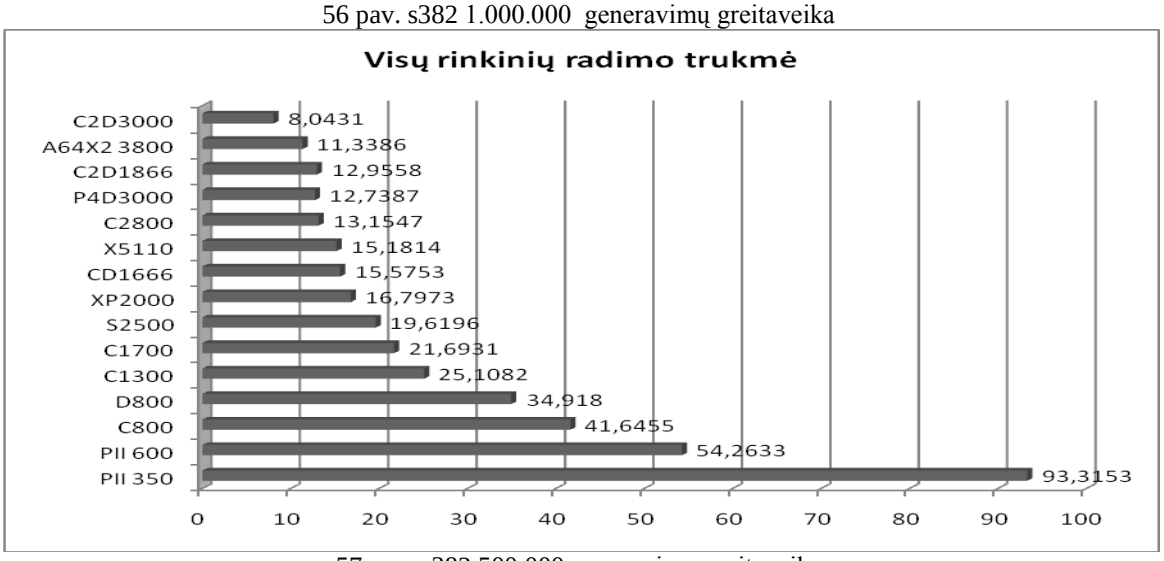
55 pav. s349 rastų K narių priklausomybė nuo generavimų skaičiaus

ISCAS-89 s382

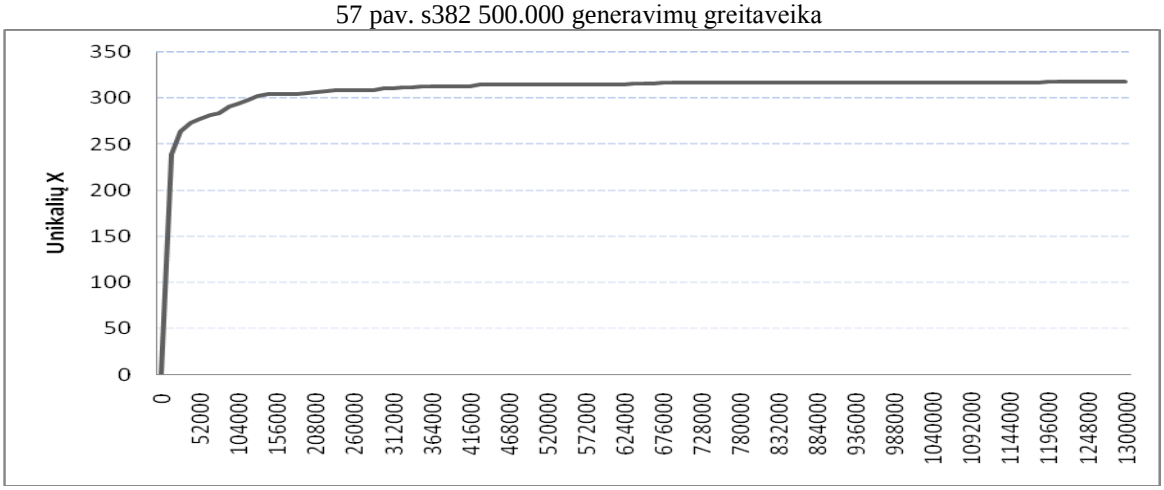
Statistika: 3 įėjimai; 6 išėjimai; 99 ventiliai (11 AND + 30 NAND + 24 OR + 34 NOR); 59 inverteriai; 21 trigeris



56 pav. s382 1.000.000 generavimų greitaveika



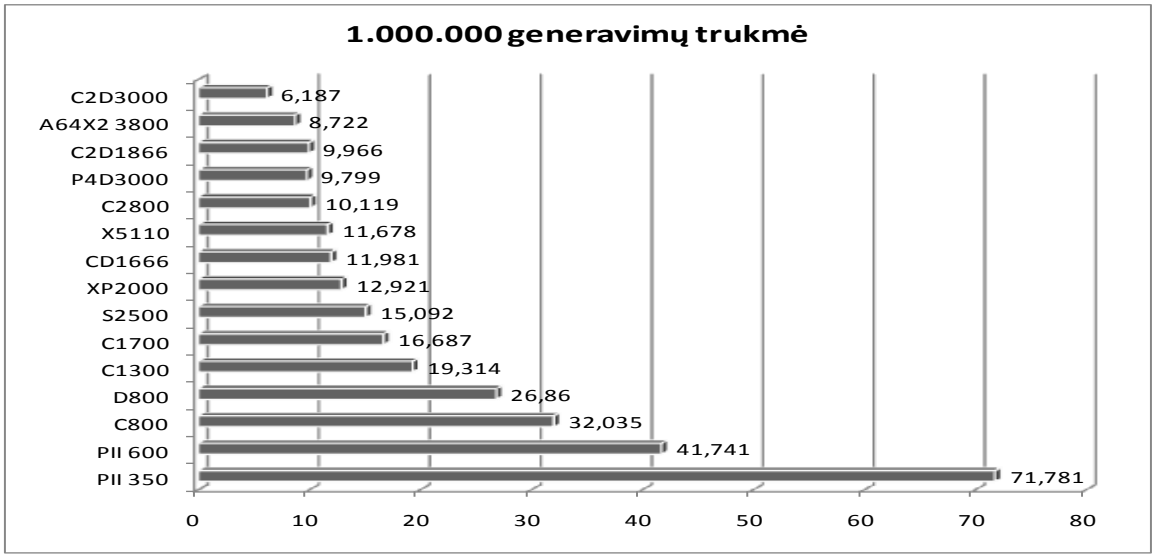
57 pav. s382 500.000 generavimų greitaveika



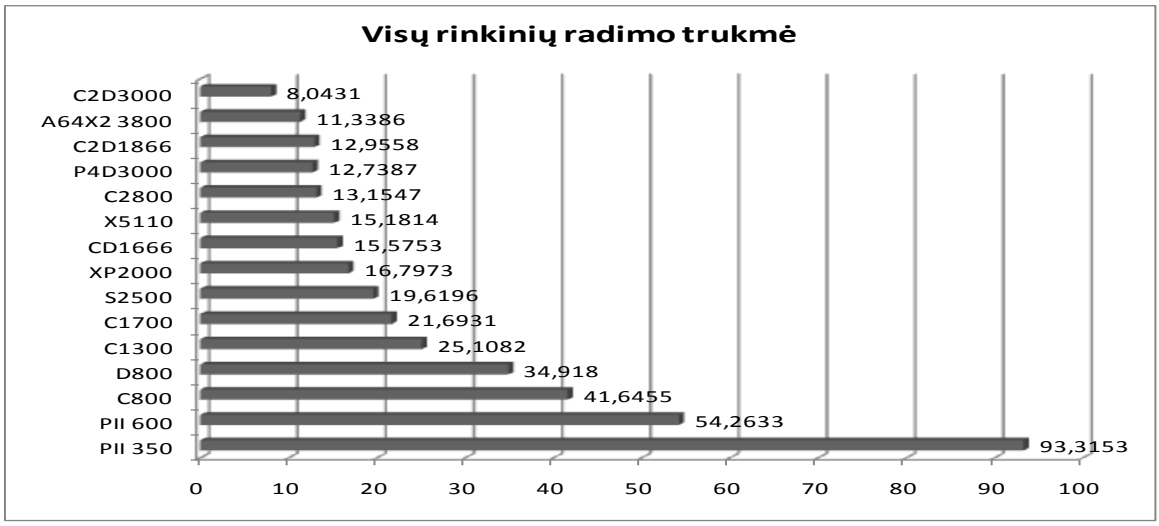
58 pav. s382 rastų K narių priklausomybė nuo generavimų skaičiaus

ISCAS-89 s386

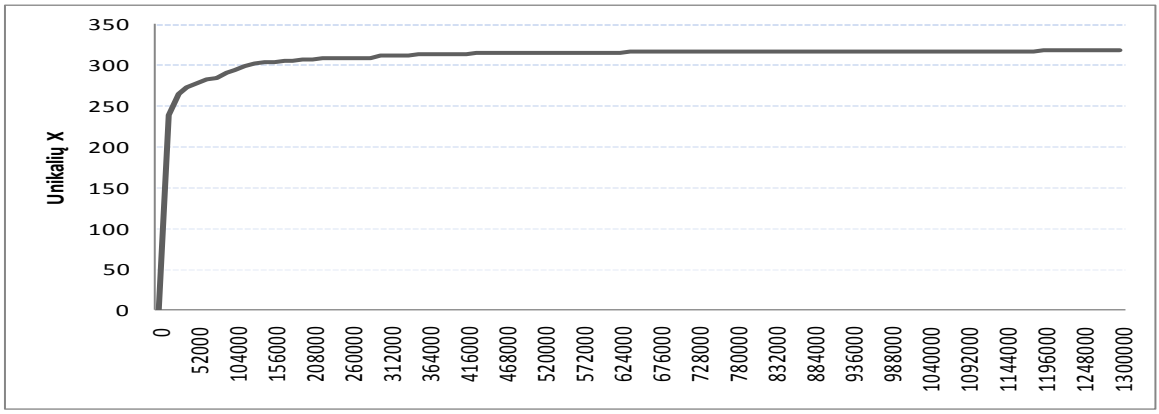
Statistika: 7 įėjimai; 7 išėjimai; 118 ventilių (83 AND + 0 NAND + 35 OR + 0 NOR); 41 inverteris; 6 trigeriai



59 pav. s386 1.000.000 generavimų greitaveika



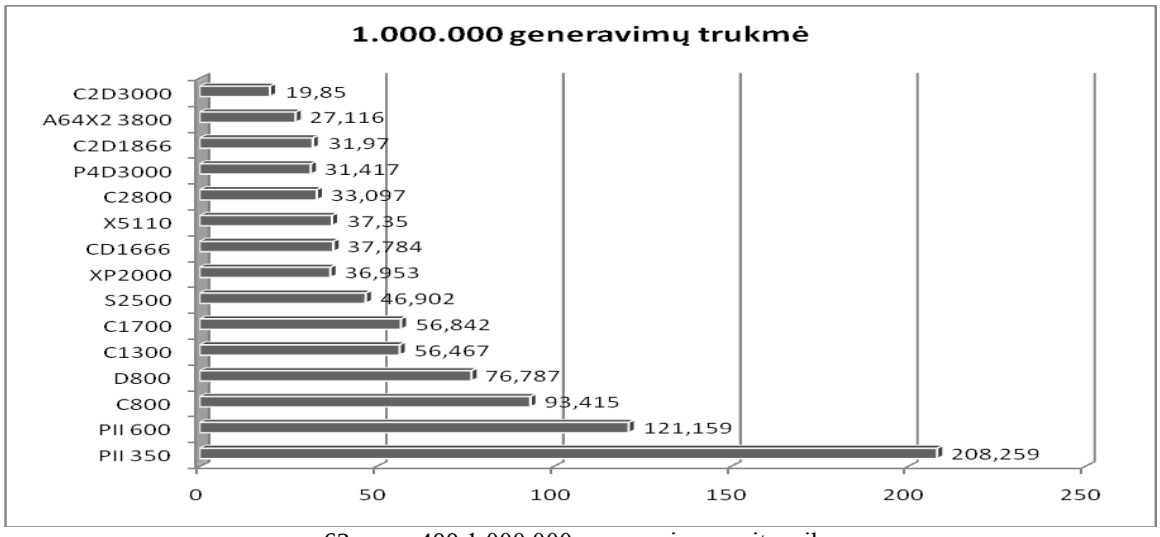
60 pav. s386 1.300.000 generavimų greitaveika



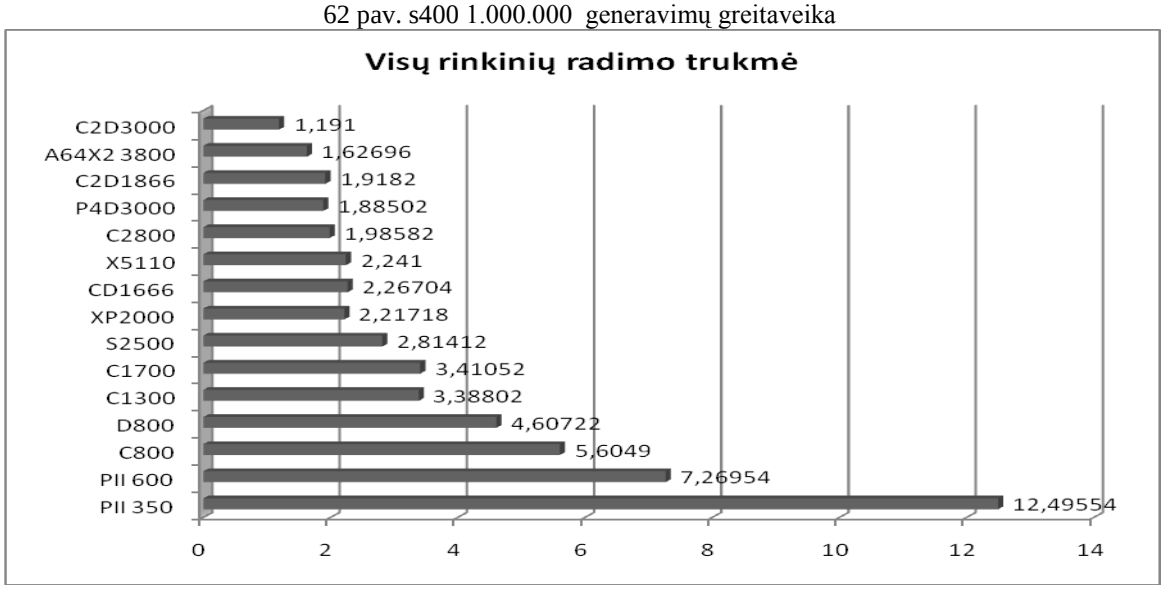
61 pav. s386 rastų K narių priklausomybė nuo generavimų skaičiaus

ISCAS-89 s400

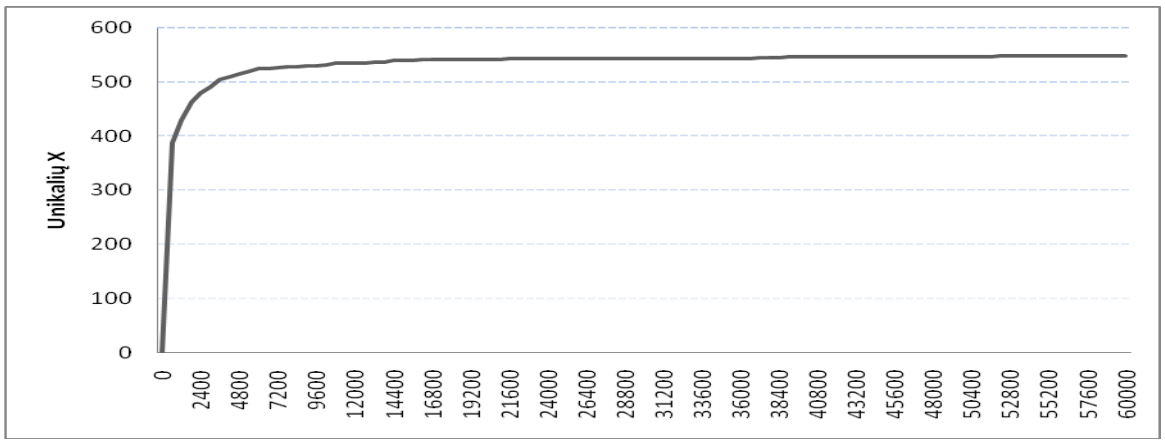
Statistika: 3 įėjimai; 6 išėjimai; 106 ventiliai (11 AND + 36 NAND + 25 OR + 34 NOR); 56 inverteriai; 21 trigeris



62 pav. s400 1.000.000 generavimų greitaveika



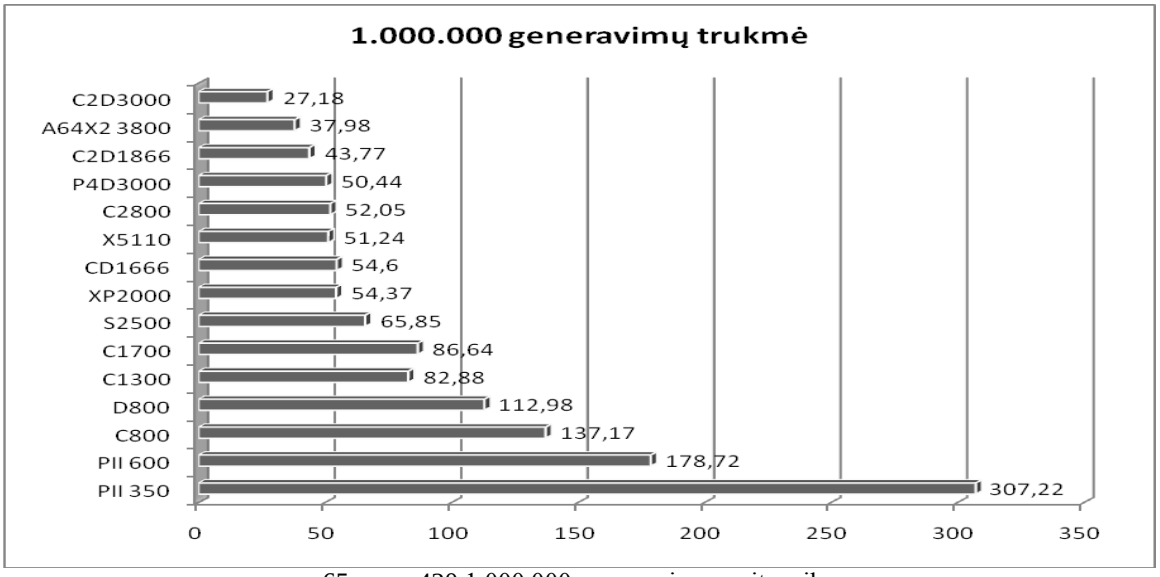
63 pav. s400 60.000 generavimų greitaveika



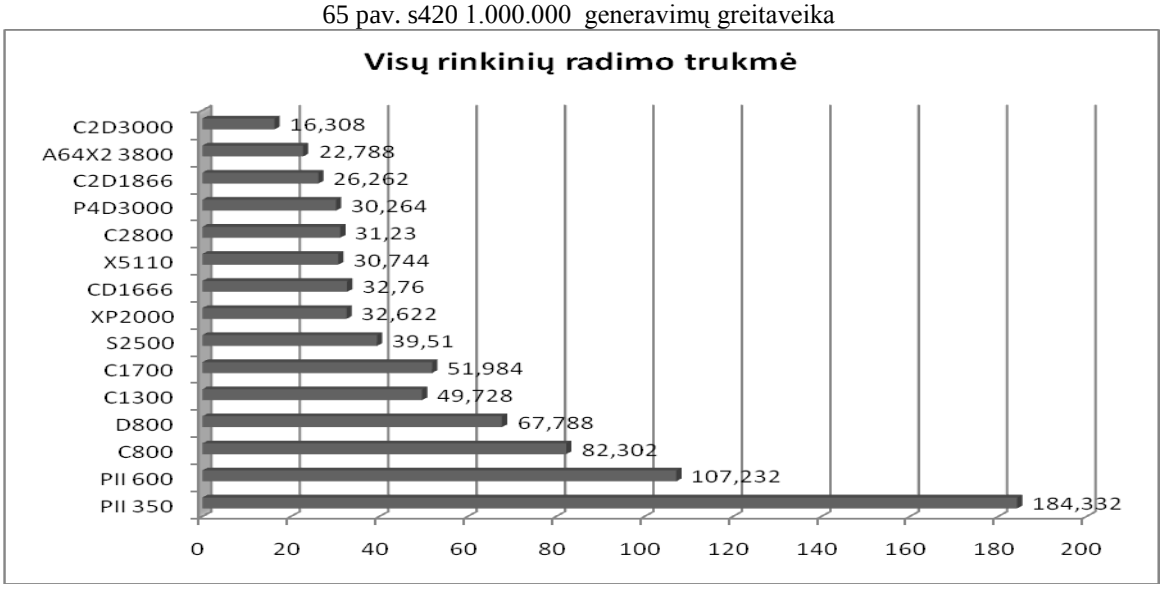
64 pav. s400 rastų K narių priklausomybė nuo generavimų skaičiaus

ISCAS-89 s420

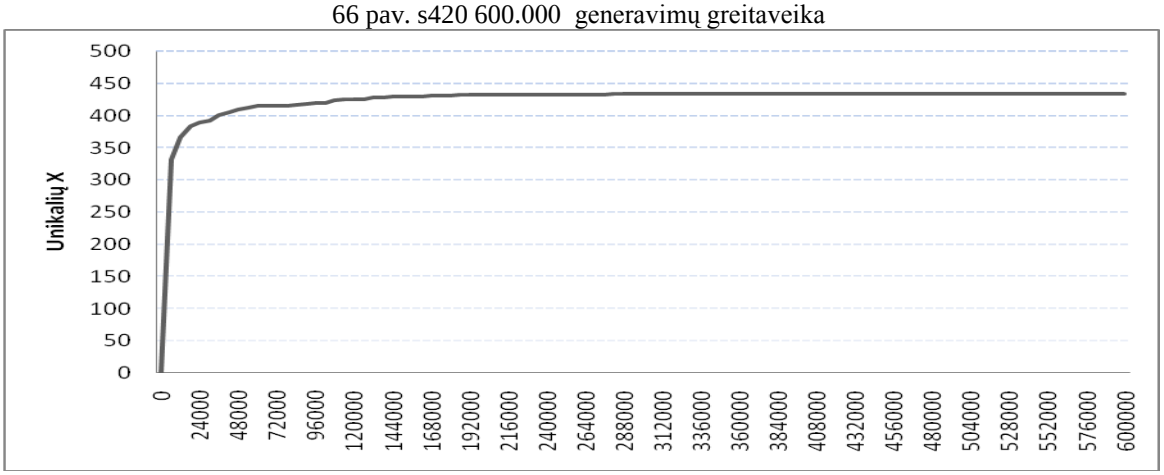
Statistika: 19 įėjimų; 2 išėjimai; 122 ventiliai (28 AND + 46 NAND + 9 OR + 39 NOR); 74 inverteriai; 16 trigerių



65 pav. s420 1.000.000 generavimų greitaveika



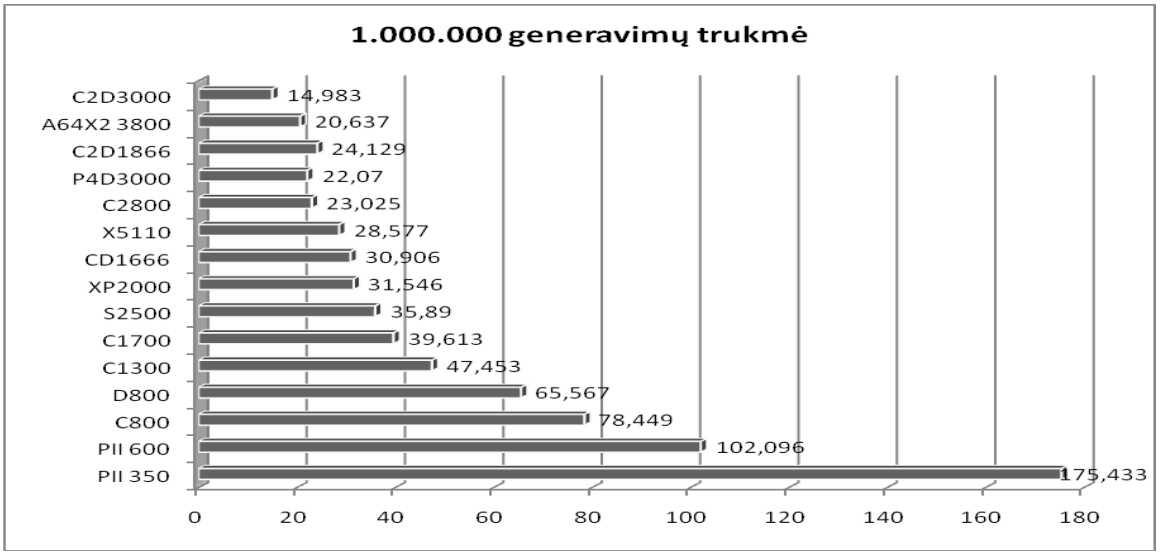
66 pav. s420 600.000 generavimų greitaveika



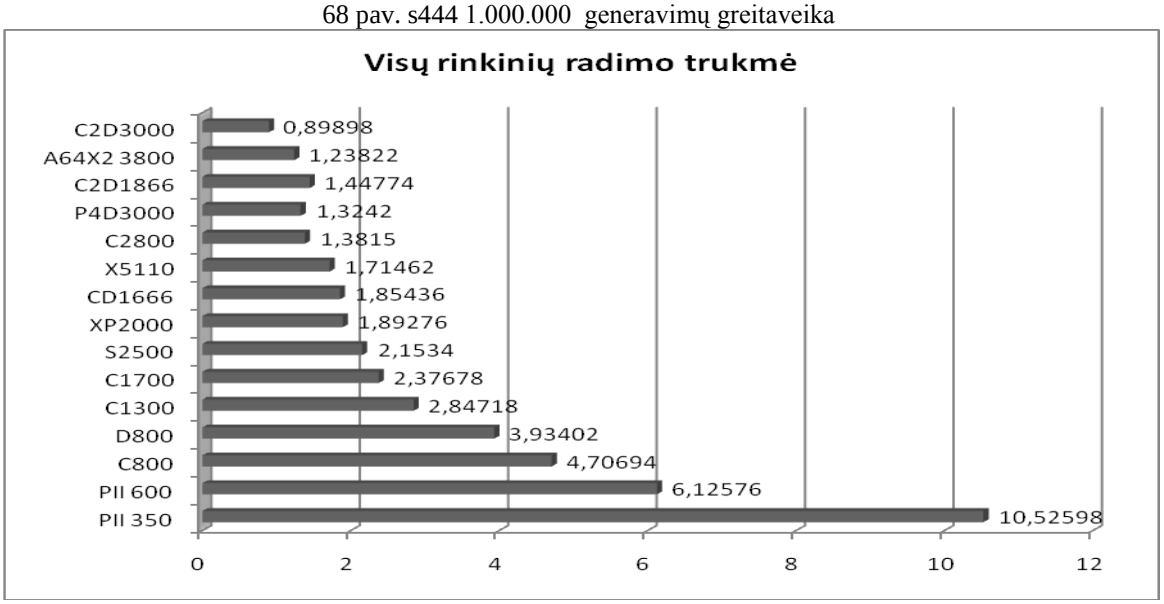
67 pav. s420 rastų K narių priklausomybė nuo generavimų skaičiaus

ISCAS-89 s444

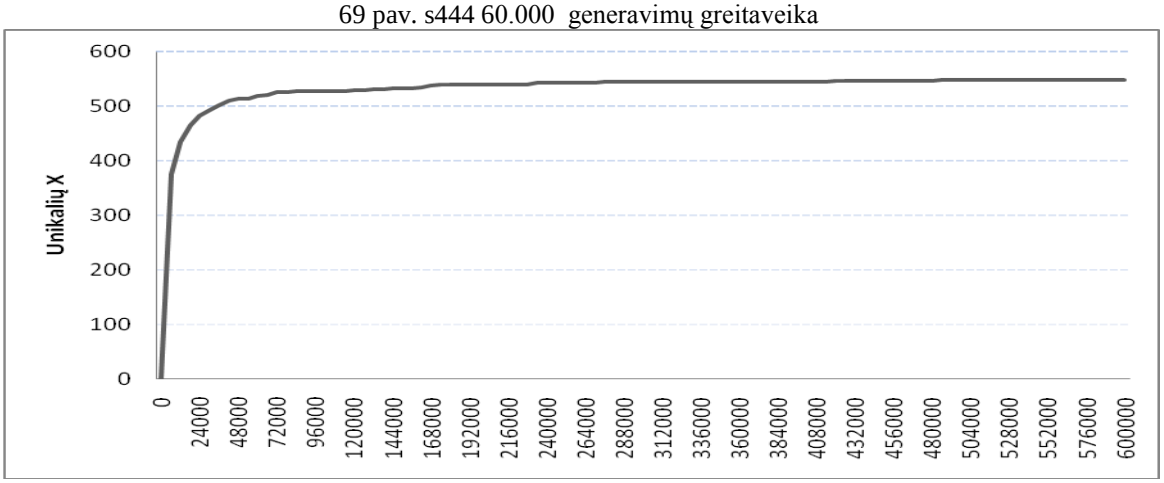
Statistika: 3 įėjimai; 6 išėjimai; 119 ventilių (13 AND + 58 NAND + 14 OR + 34 NOR); 62 inverteriai; 21 trigeris



68 pav. s444 1.000.000 generavimų greitaveika



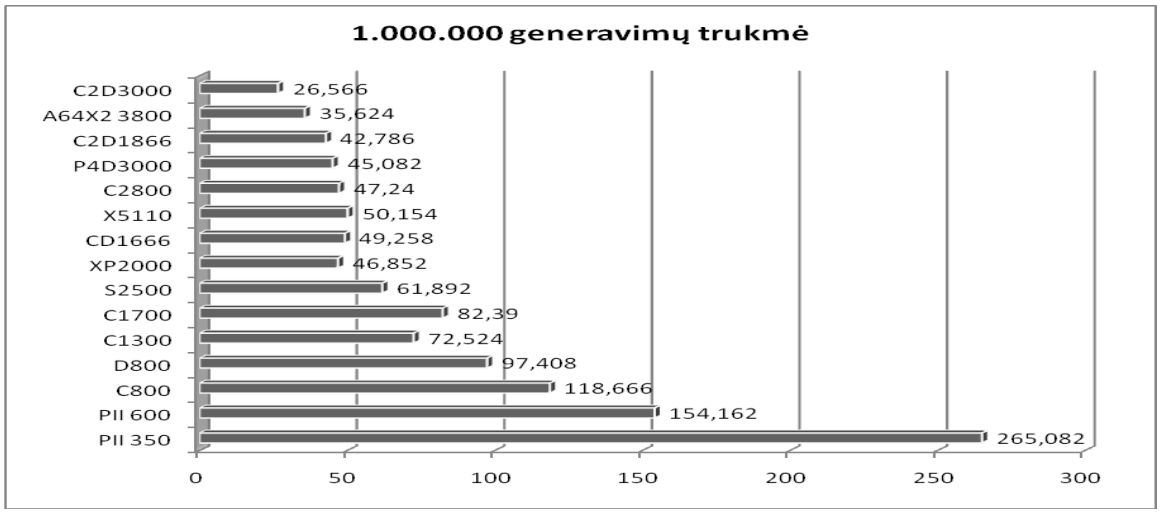
69 pav. s444 60.000 generavimų greitaveika



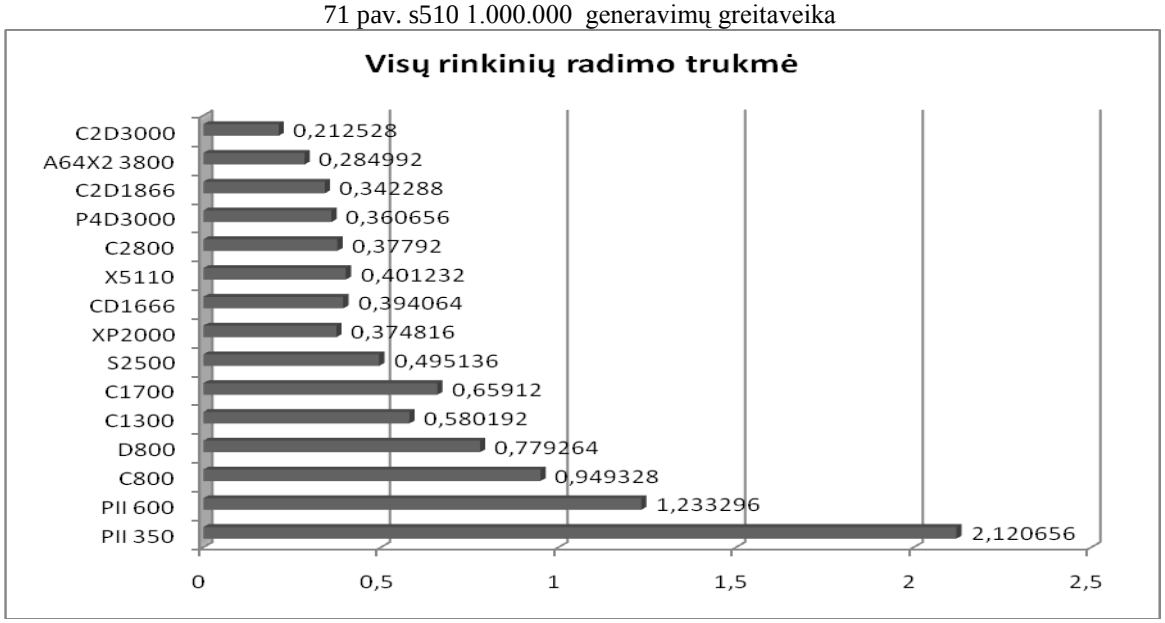
70 pav. s444 rastų K narių priklausomybė nuo generavimų skaičiaus

ISCAS-89 s510

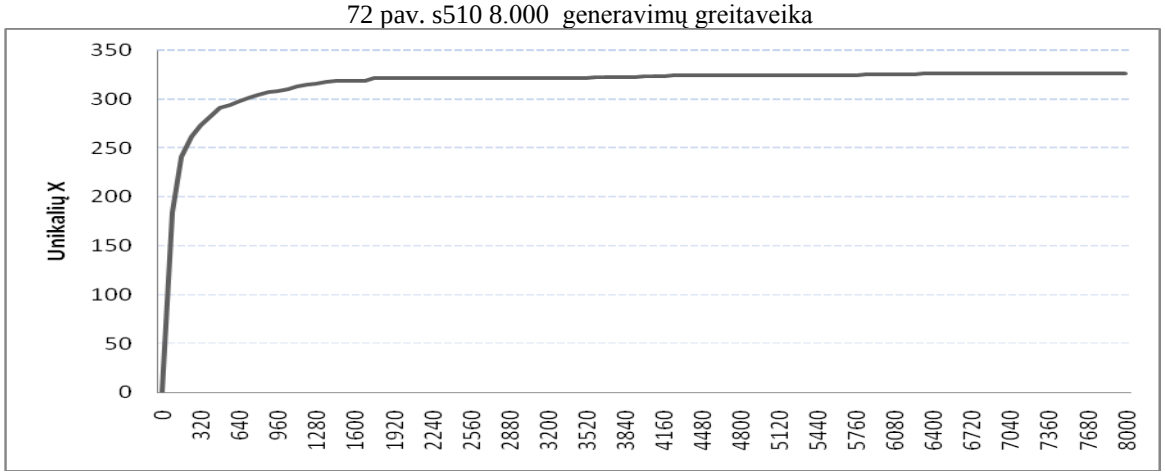
Statistika: 19 įėjimų; 7 išėjimai; 179 ventiliai (34 AND + 61 NAND + 29 OR + 55 NOR); 32 inverteriai; 6 trigeriai



71 pav. s510 1.000.000 generavimų greitaveika



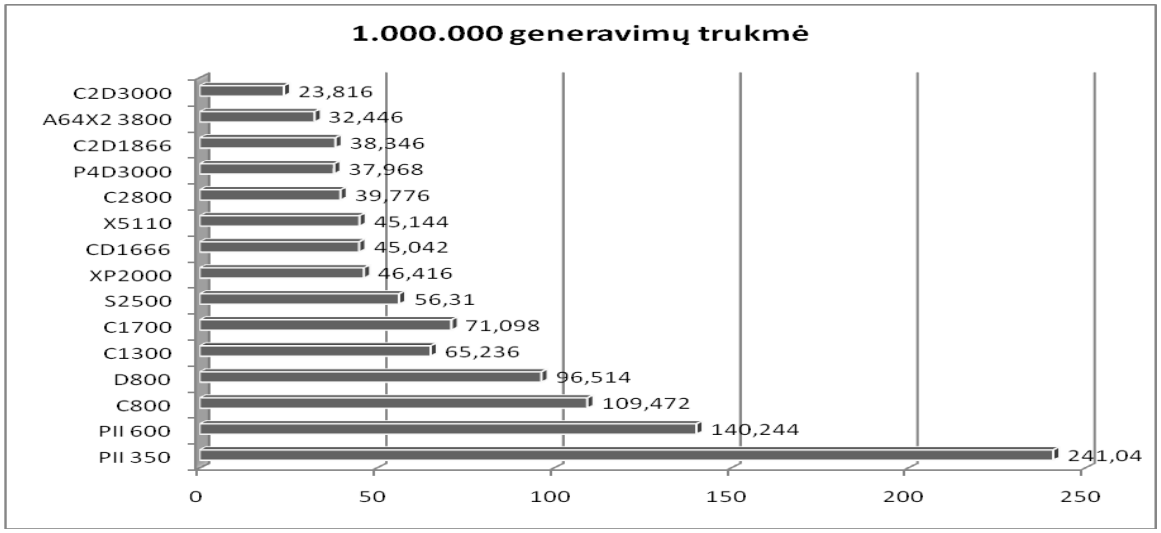
72 pav. s510 8.000 generavimų greitaveika



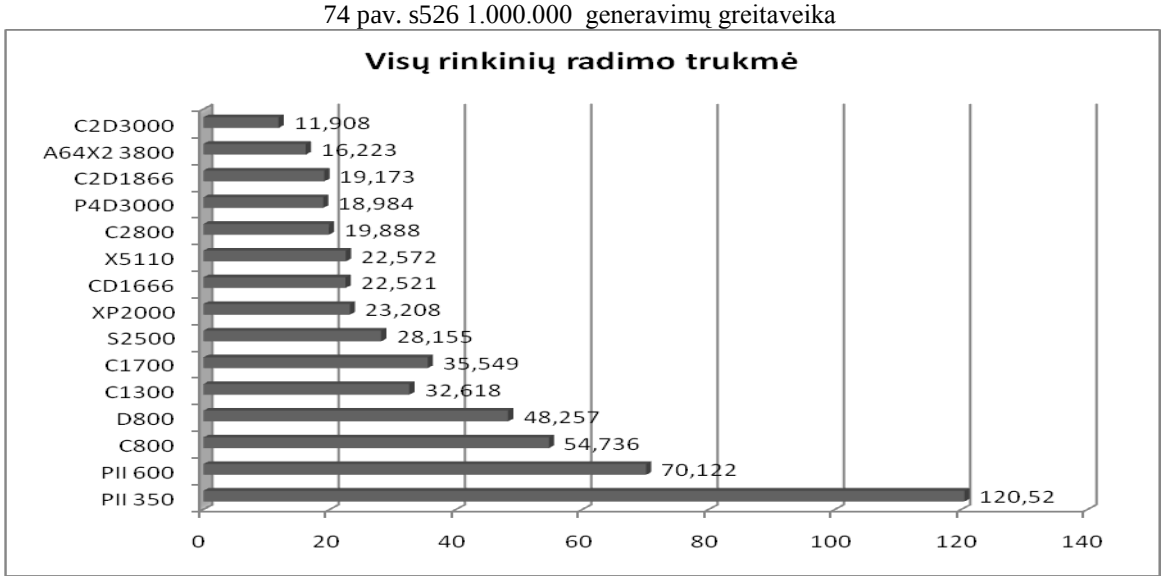
73 pav. s510 rastų K narių priklausomybė nuo generavimų skaičiaus

ISCAS-89 s526

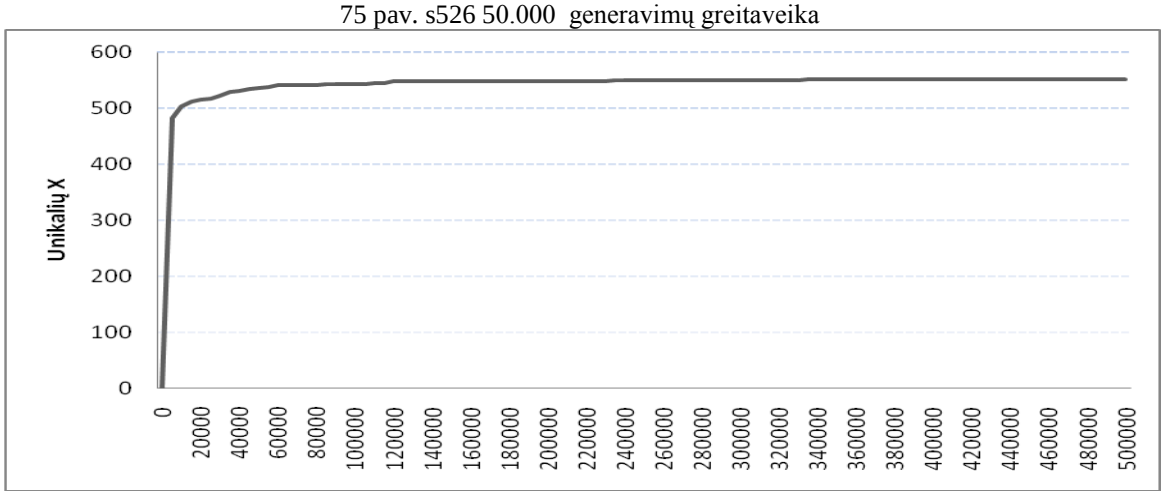
Statistika: 3 įėjimai; 6 išėjimai; 141 ventiliai (56 AND + 22 NAND + 28 OR + 35 NOR); 52 inverteriai; 21 trigeris



74 pav. s526 1.000.000 generavimų greitaveika



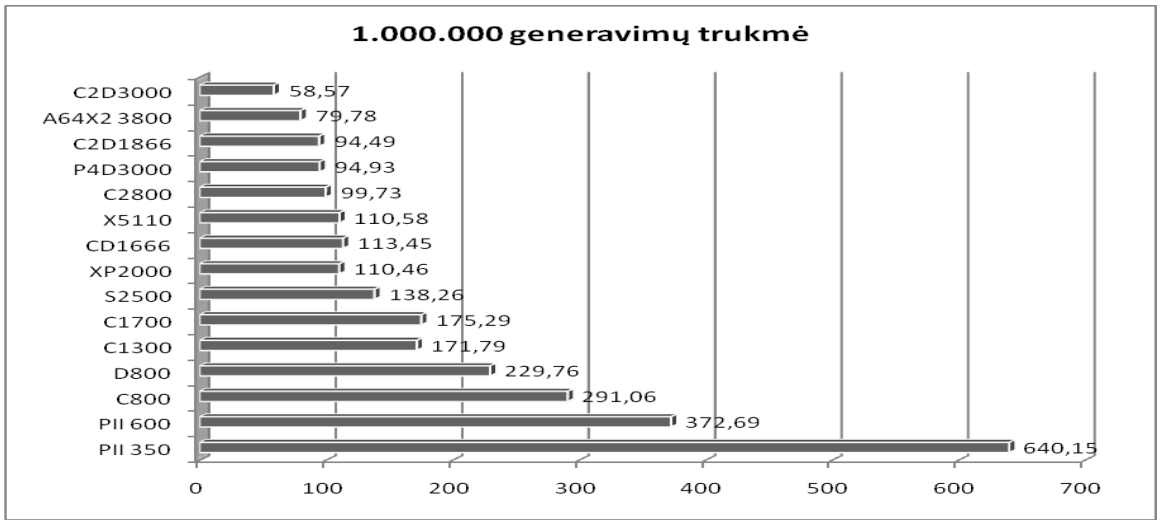
75 pav. s526 50.000 generavimų greitaveika



76 pav. s526 rastų K narių priklausomybė nuo generavimų skaičiaus

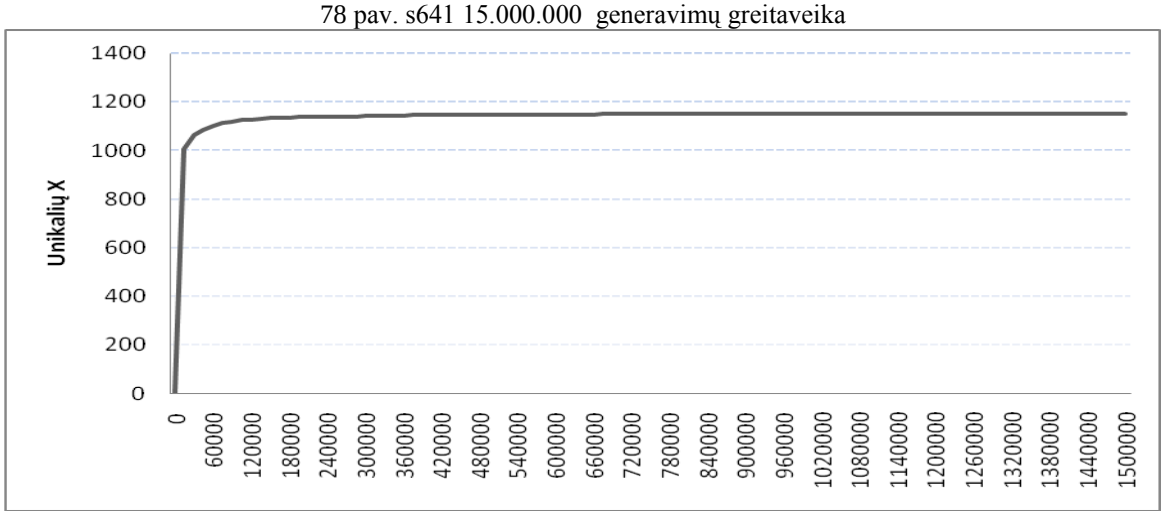
ISCAS-89 s641

Statistika: 35 įėjimai; 24 išėjimai; 107 ventiliai (90 AND + 4 NAND + 13 OR + 0 NOR); 272 inverteriai; 19 trigerių



77 pav. s641 1.000.000 generavimų greitaveika

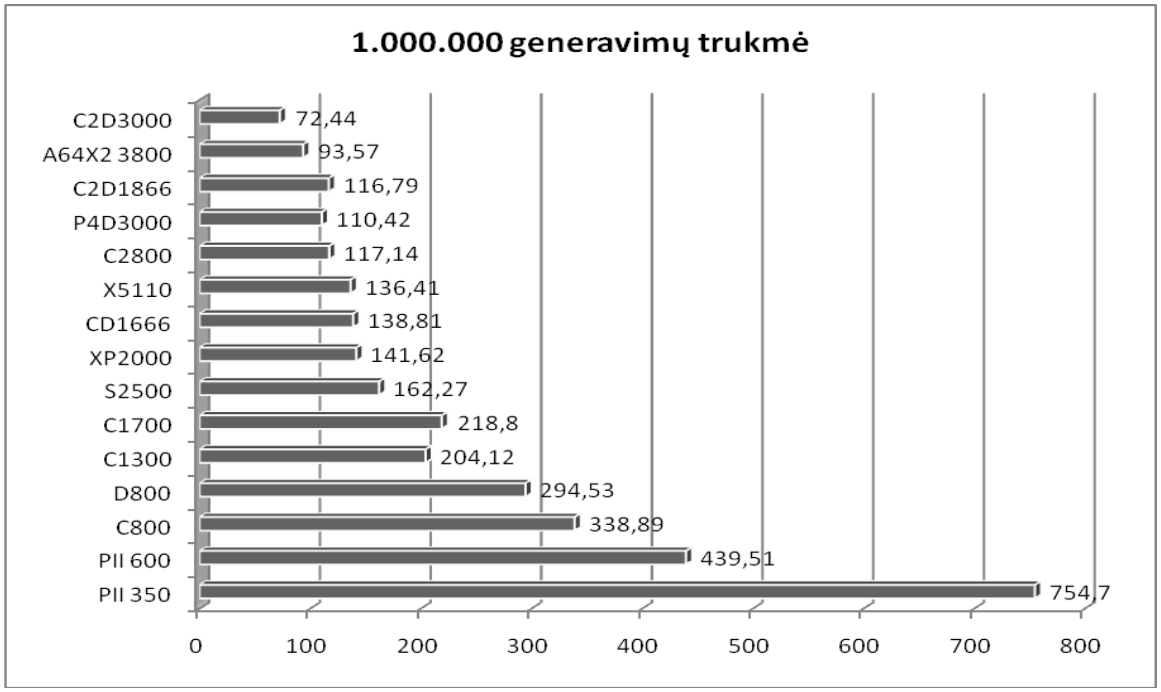
78 pav. s641 15.000.000 generavimų greitaveika



79 pav. rastų K narių priklausomybė nuo generavimų skaičiaus

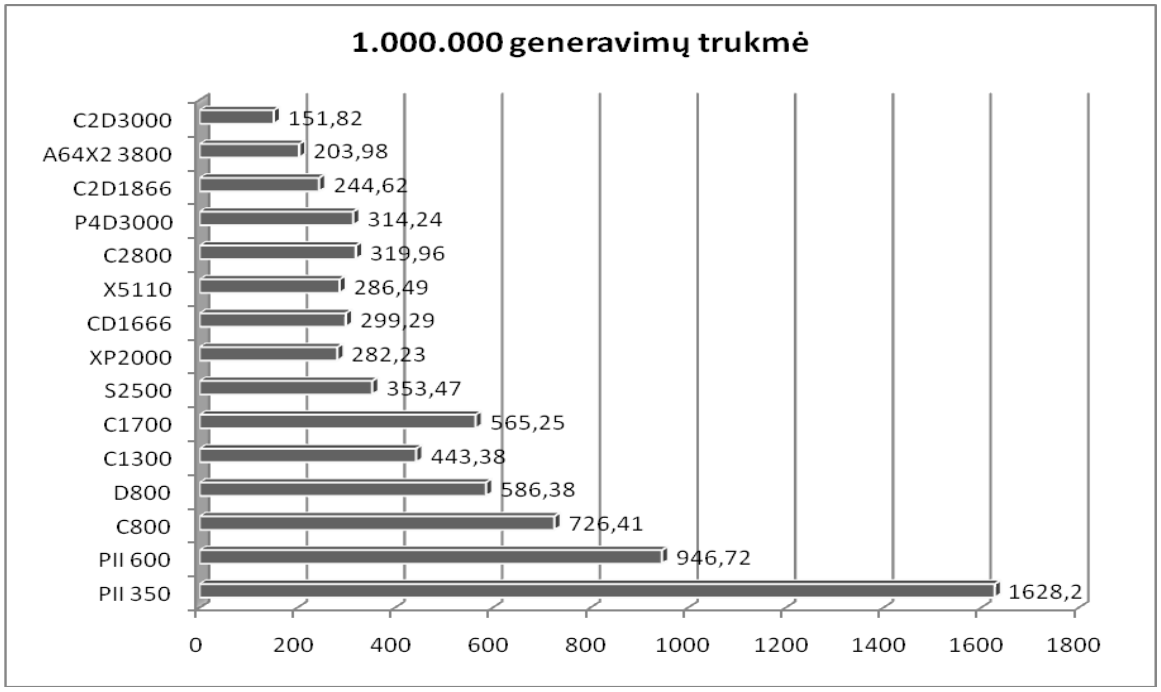
ISCAS-89 s713 ir s838

s713 Statistika: 35 įėjimai; 23 išėjimai; 139 ventiliai (94 AND + 28 NAND + 17 OR + 0 NOR); 254 inverteriai; 19 trigerių



80 pav. s713 1.000.000 generavimų greitaveika

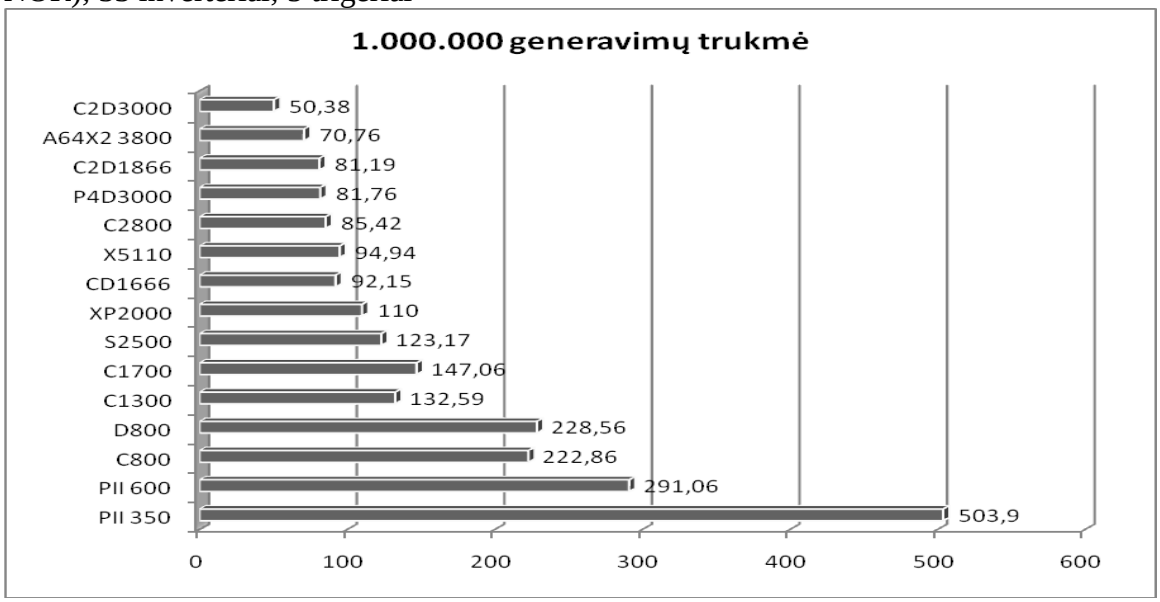
s838 Statistika: 35 įėjimai; 2 išėjimai; 241 ventiliai (58 AND + 89 NAND + 16 OR + 78 NOR); 149 inverteriai; 32 trigeriai



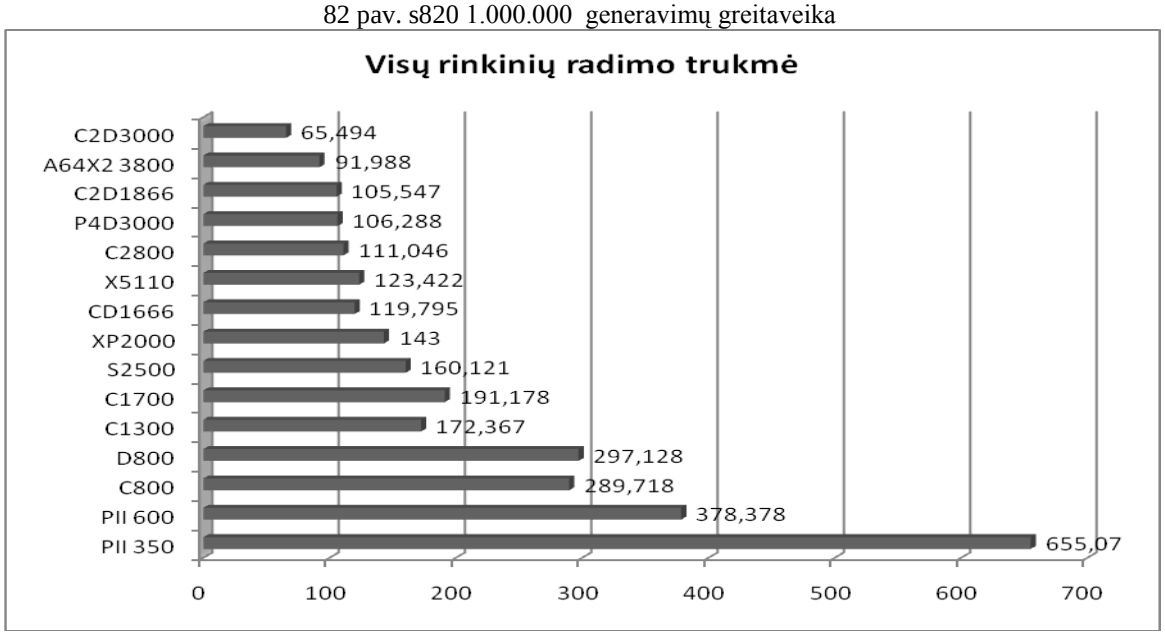
81 pav. s383 1.000.000 generavimų greitaveika

ISCAS-89 s820

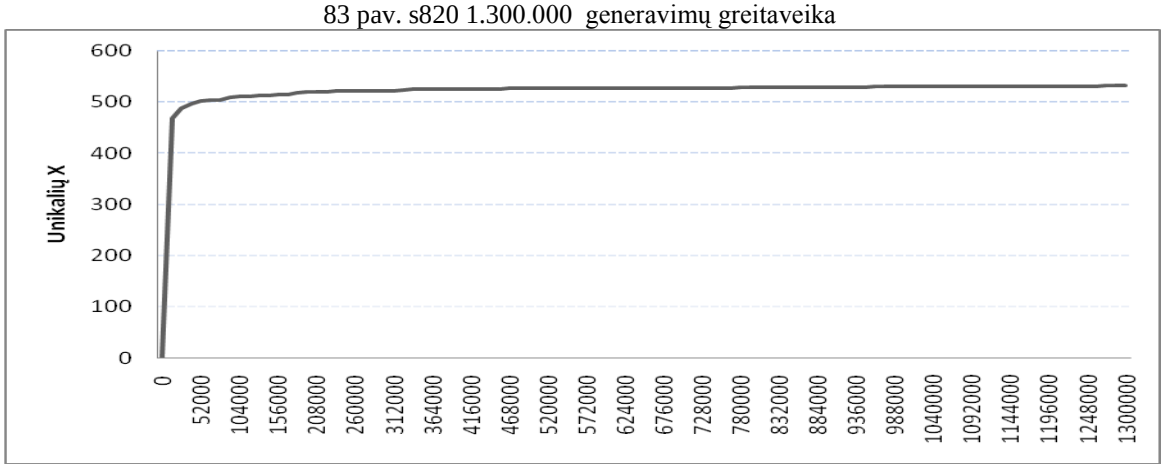
Statistika: 18 įėjimų; 19 išėjimų; 256 ventiliai (76 AND + 54 NAND + 60 OR + 66 NOR); 33 inverteriai; 5 trigeriai



82 pav. s820 1.000.000 generavimų greitaveika



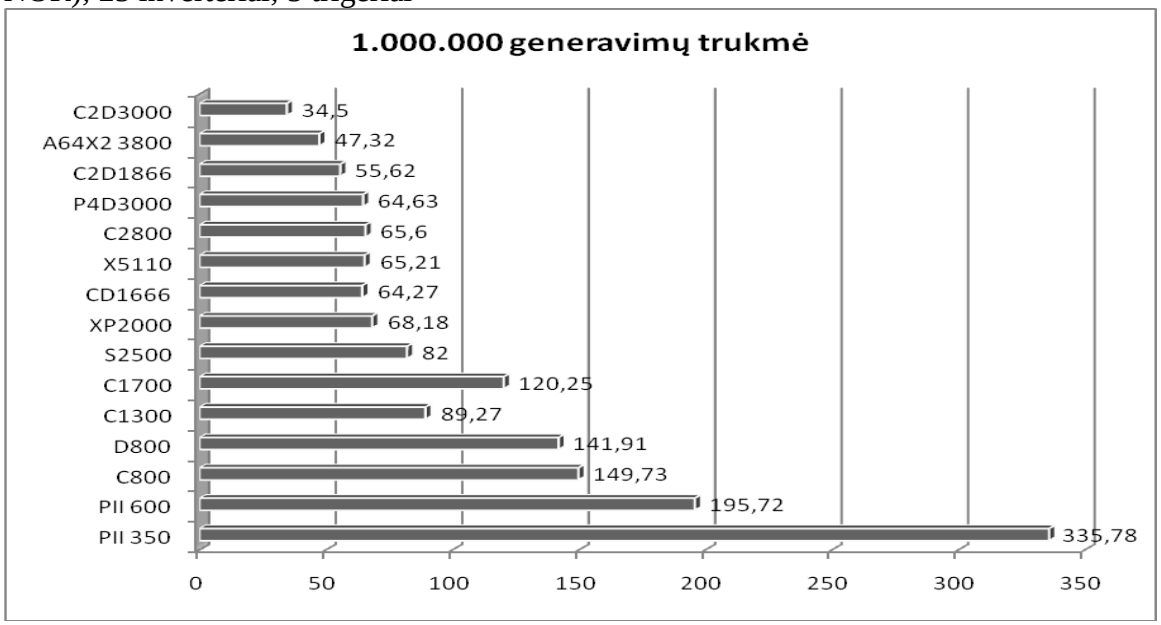
83 pav. s820 1.300.000 generavimų greitaveika



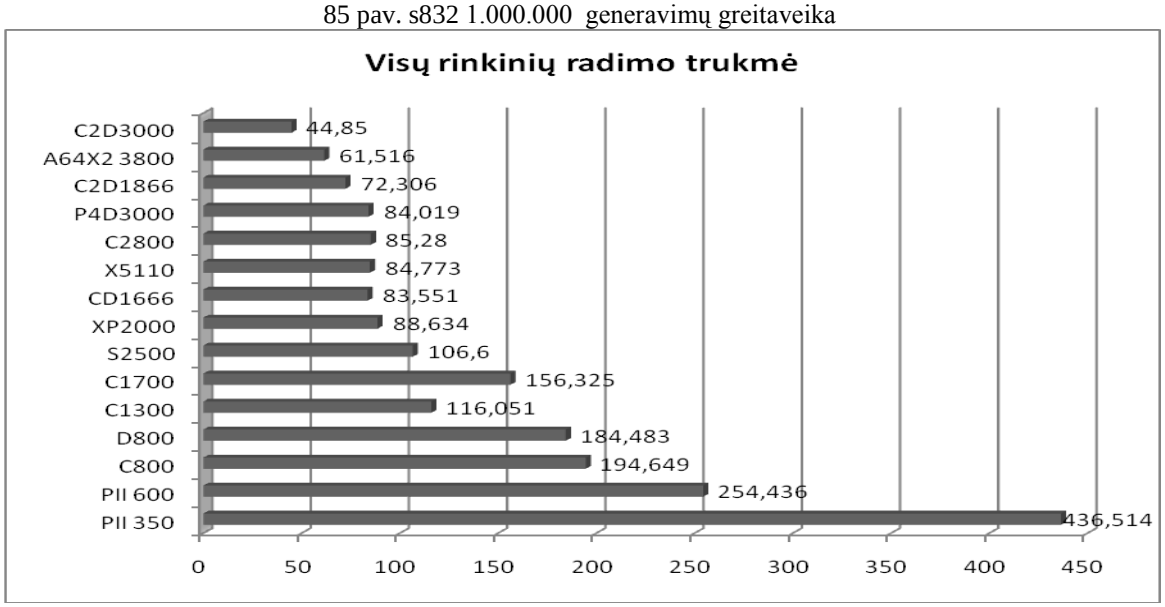
84 pav. s820 rastų K narių priklausomybė nuo generavimų skaičiaus

ISCAS-89 s832

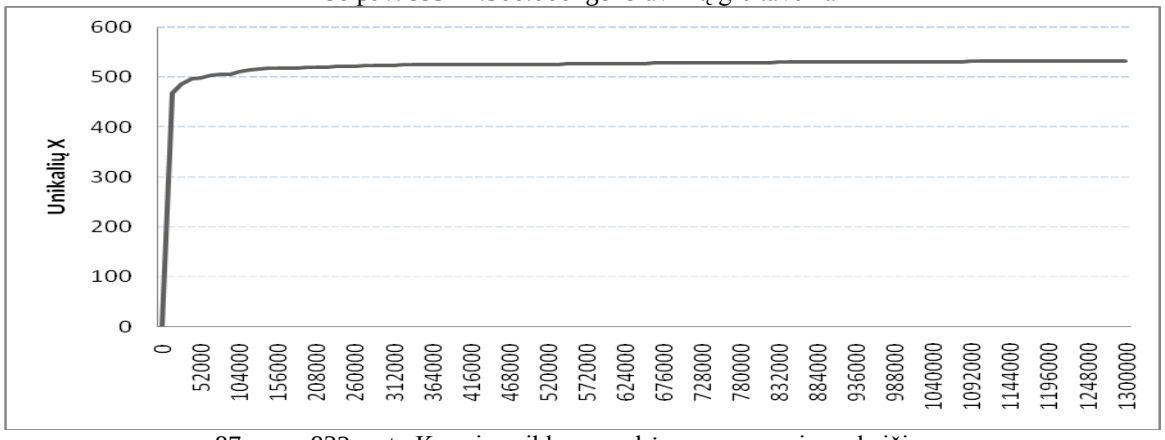
Statistika: 18 įėjimų; 19 išėjimų; 262 ventiliai (78 AND + 54 NAND + 64 OR + 66 NOR); 25 inverteriai; 5 trigeriai



85 pav. s832 1.000.000 generavimų greitaveika



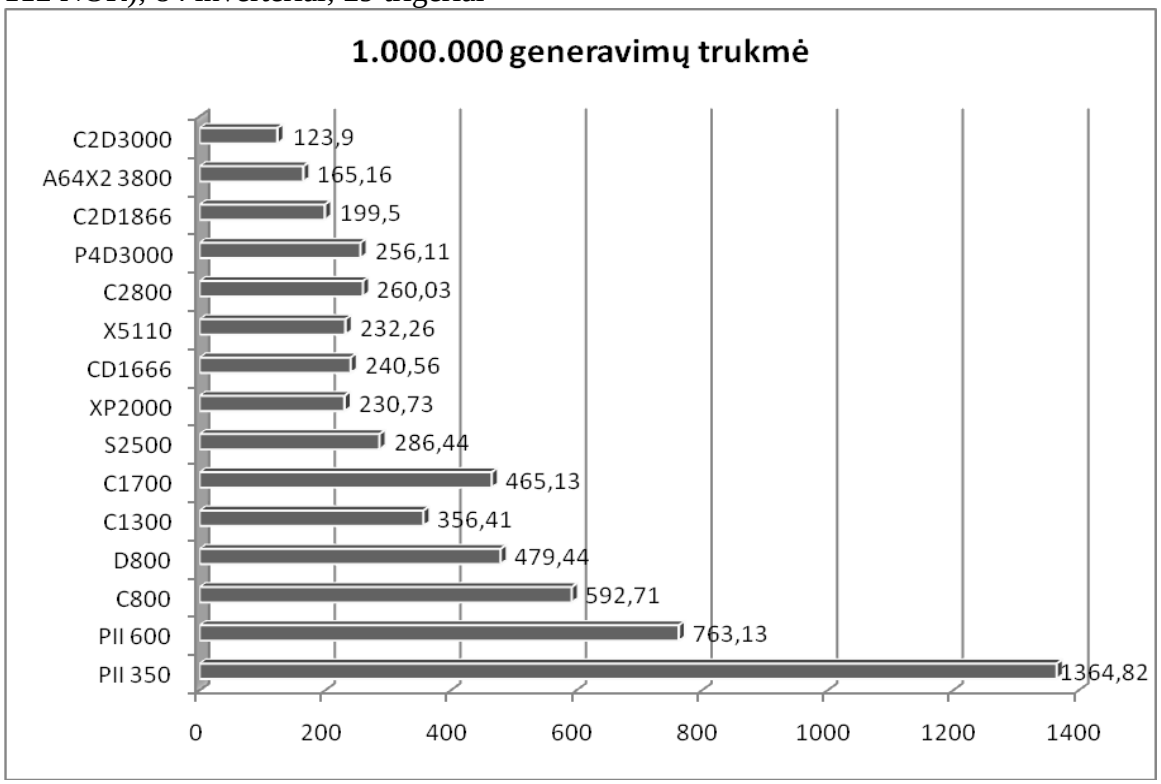
86 pav. s832 1.300.000 generavimų greitaveika



87 pav. s832 rastų K narių priklausomybė nuo generavimų skaičiaus

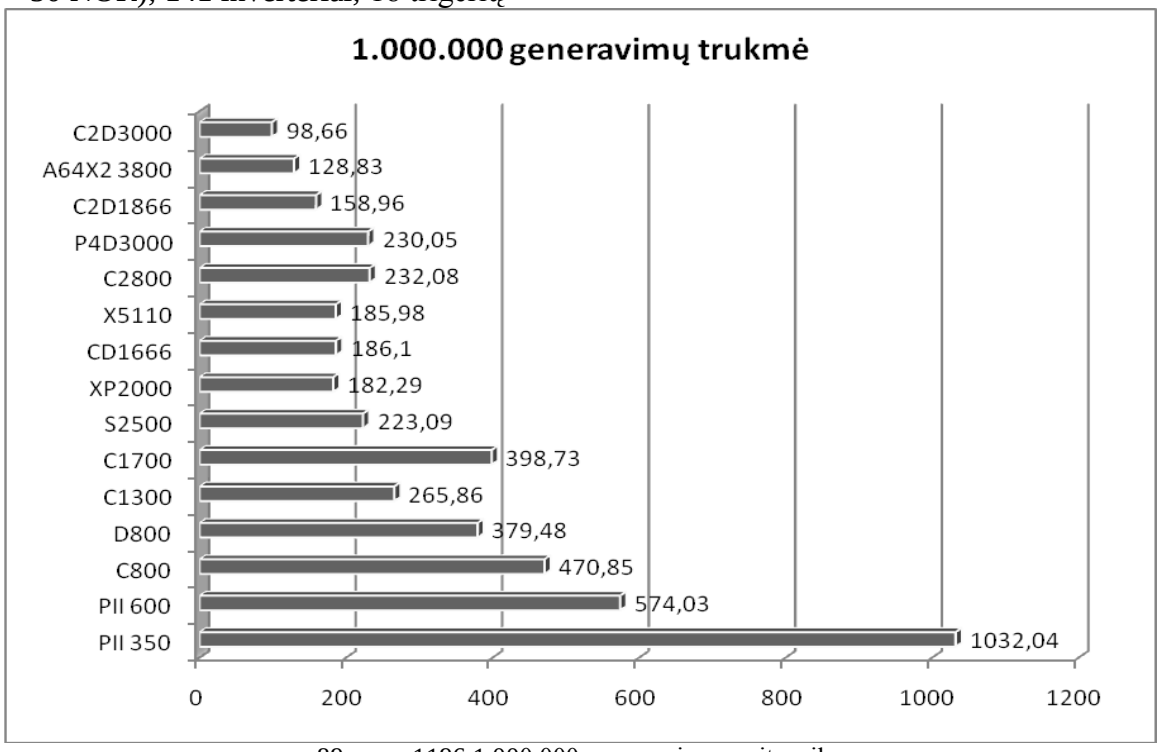
ISCAS-89 s953 ir s1196

s953 Statistika: 16 įėjimų; 23 išėjimai; 311 ventilių (49 AND + 114 NAND + 36 OR + 112 NOR); 84 inverteriai; 29 trigeriai



88 pav. s953 1.000.000 generavimų greitaveika

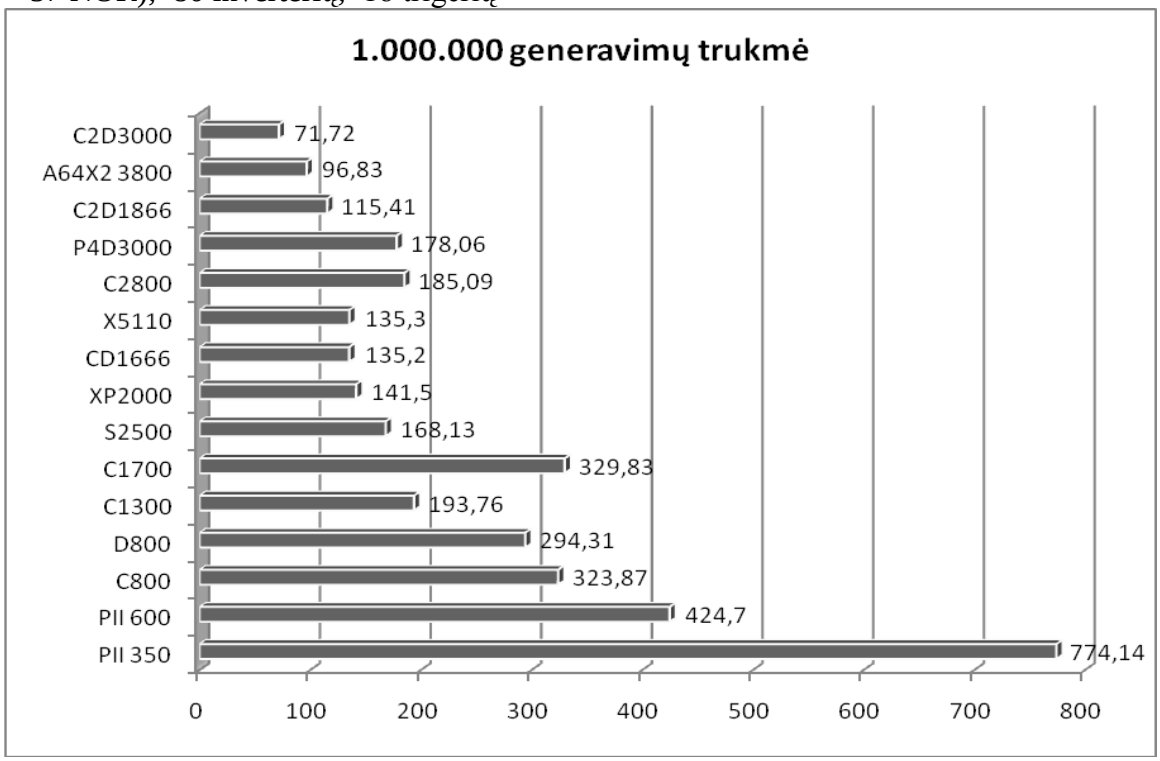
s1196 Statistika: 14 įėjimų; 14 išėjimų; 388 ventiliai (118 AND + 119 NAND + 101 OR + 50 NOR); 141 inverteriai; 18 trigerių



89 pav. s1196 1.000.000 generavimų greitaveika

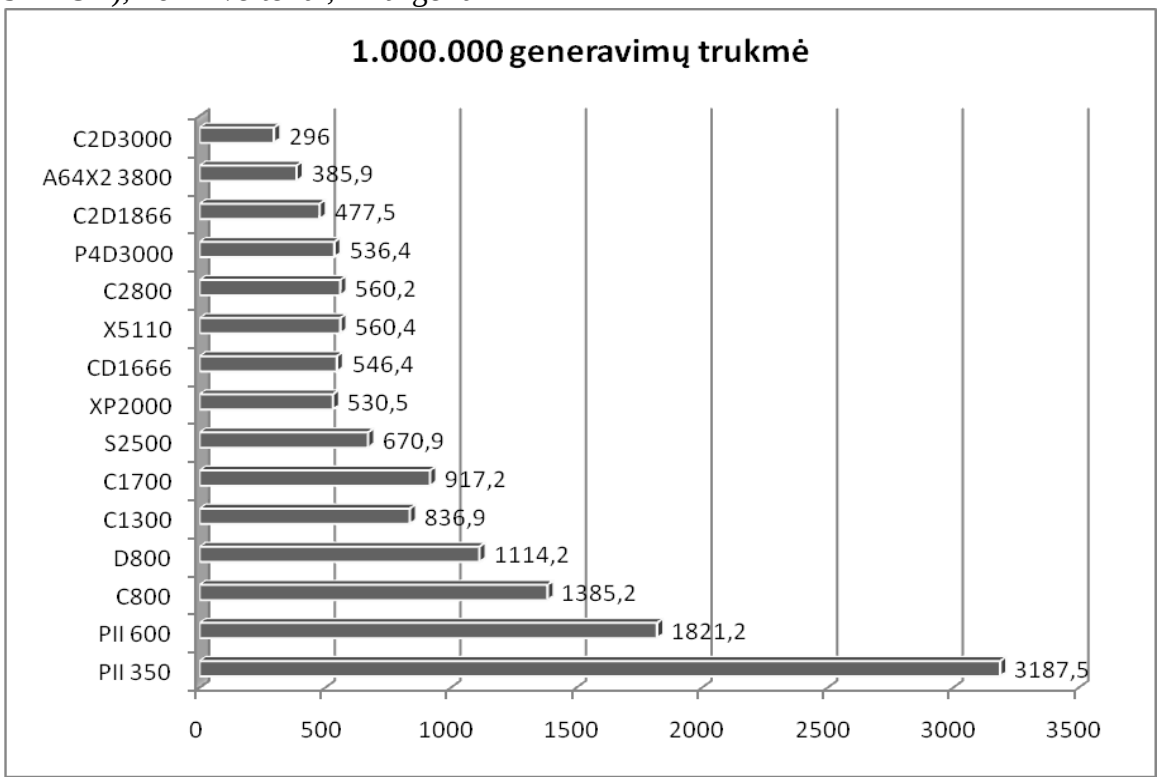
ISCAS-89 s1238 ir s1423

s1238 Statistika: 14 įėjimų; 14 išėjimų; 428 ventiliai (134 AND + 125 NAND + 112 OR + 57 NOR); 80 inverterių; 18 trigerių



90 pav. s1238 1.000.000 generavimų greitaveika

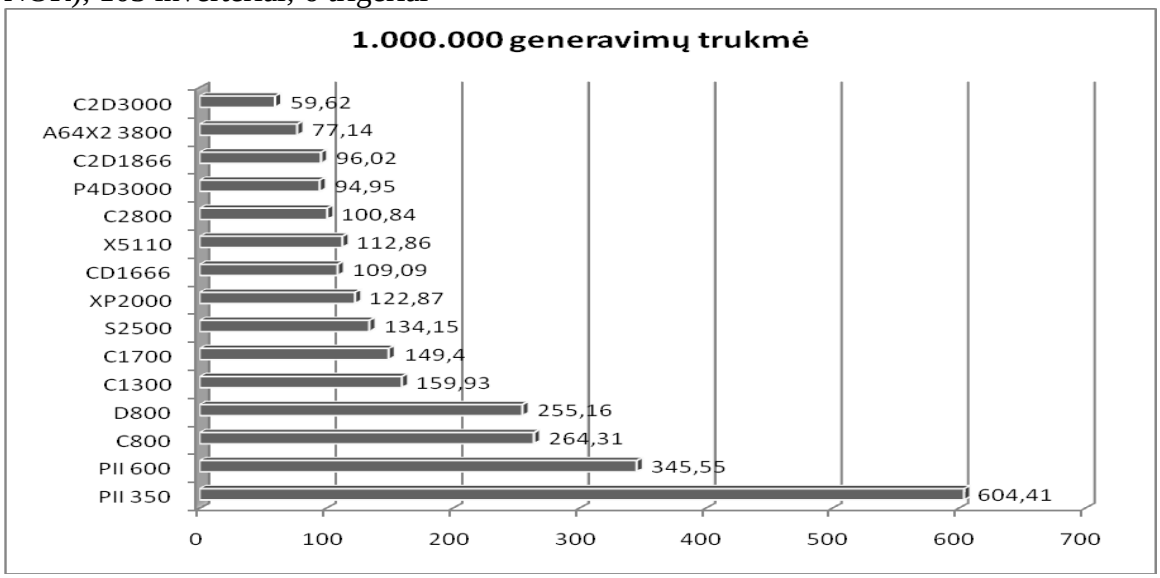
s1423 Statistika: 17 įėjimų; 5 išėjimai; 490 ventilių (197 AND + 64 NAND + 137 OR + 92 NOR); 167 inverteriai; 74 trigeriai



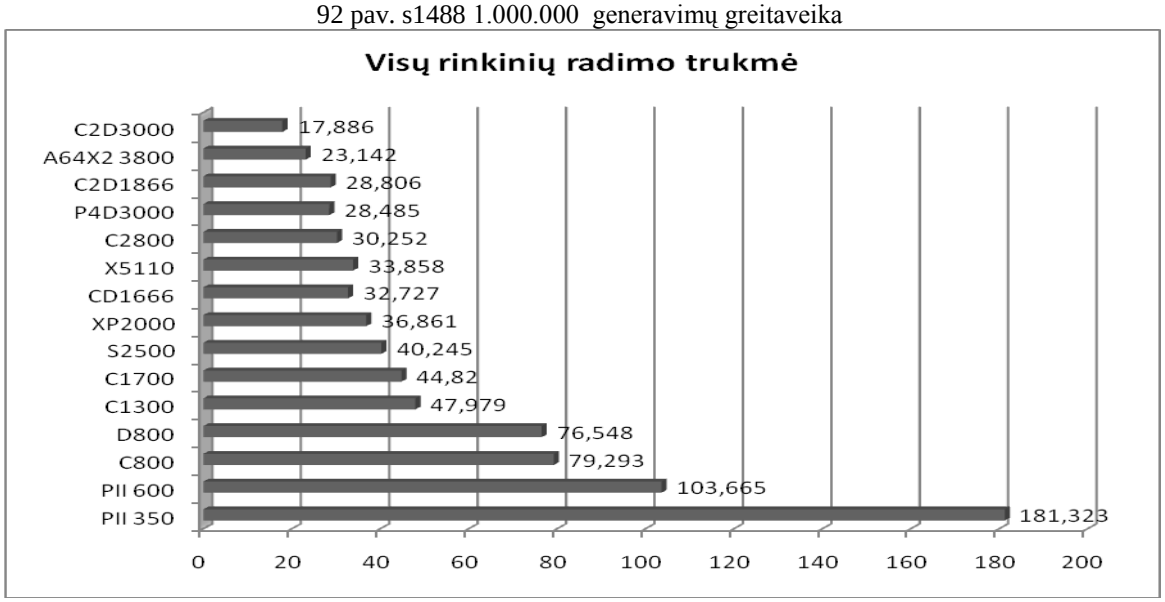
91 pav. s1423 1.000.000 generavimų greitaveika

ISCAS-89 s1488

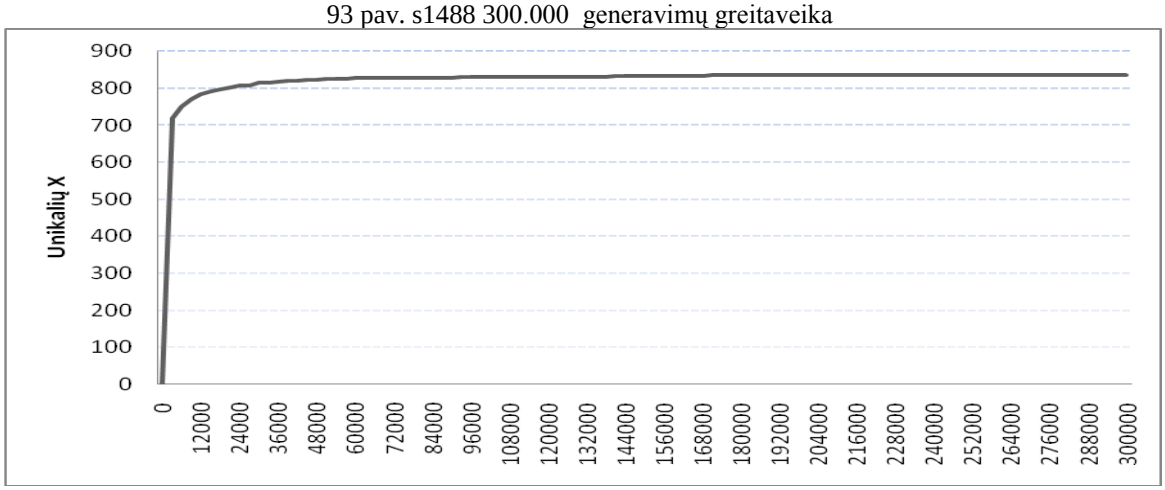
Statistika: 8 įėjimai; 19 išėjimų; 550 ventilių (350 AND + 0 NAND + 200 OR + 0 NOR); 103 inverteriai; 6 trigeriai



92 pav. s1488 1.000.000 generavimų greitaveika



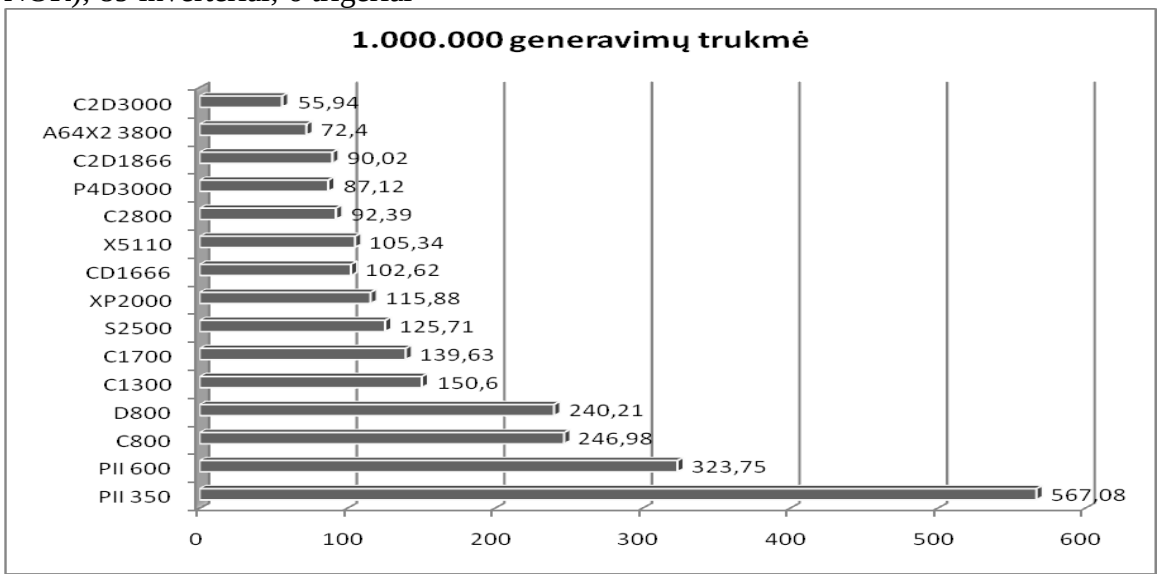
93 pav. s1488 300.000 generavimų greitaveika



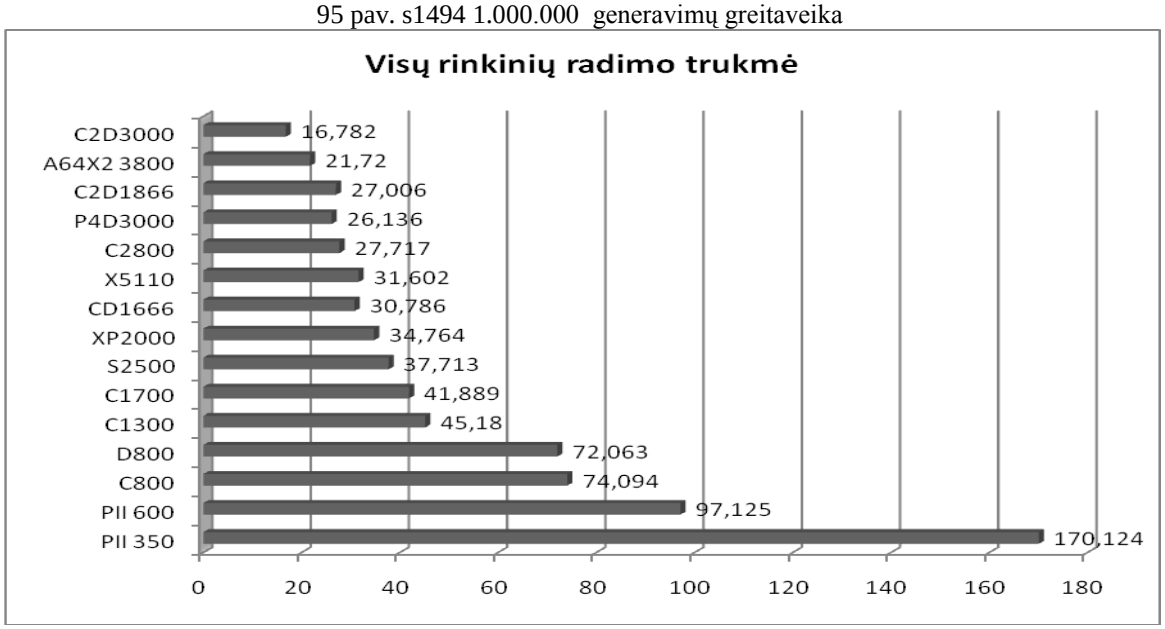
94 pav. s1488 rastų K narių priklausomybė nuo generavimų skaičiaus

ISCAS-89 s1494

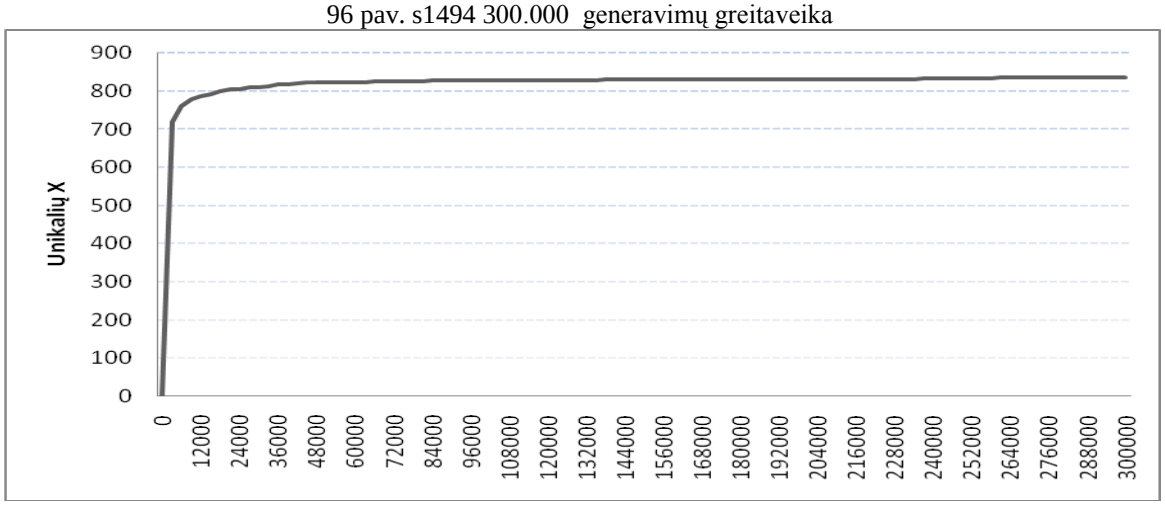
Statistika: 8 įėjimai; 19 išėjimų; 558 ventiliai (354 AND + 0 NAND + 204 OR + 0 NOR); 89 inverteriai; 6 trigeriai



95 pav. s1494 1.000.000 generavimų greitaveika



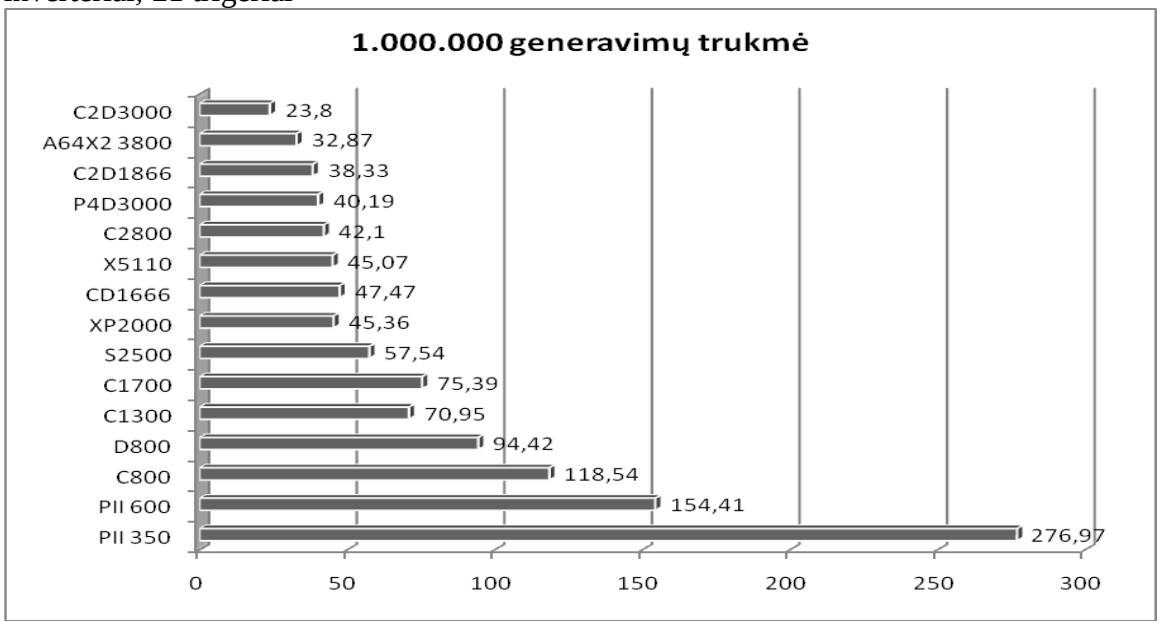
96 pav. s1494 300.000 generavimų greitaveika



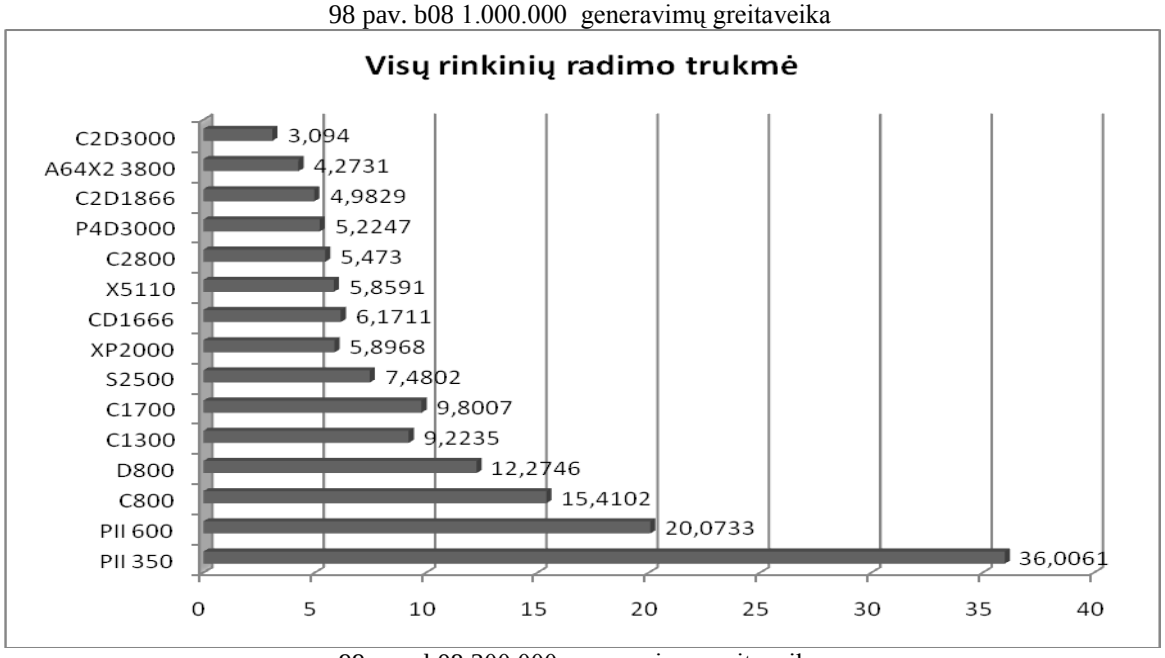
97 pav. s1494 rastų K narių priklausomybė nuo generavimų skaičiaus

ISCAS99 b08

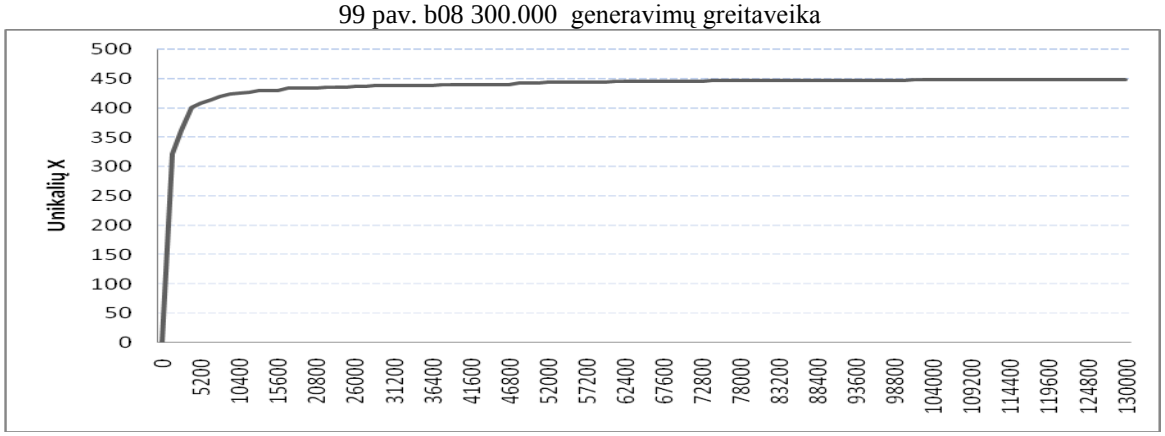
Statistika: 9 įėjimai; 4 išėjimai; 157 ventiliai (9 AND + 113 NAND + 1 OR); 26 inverteriai; 21 trigeriai



98 pav. b08 1.000.000 generavimų greitaveika



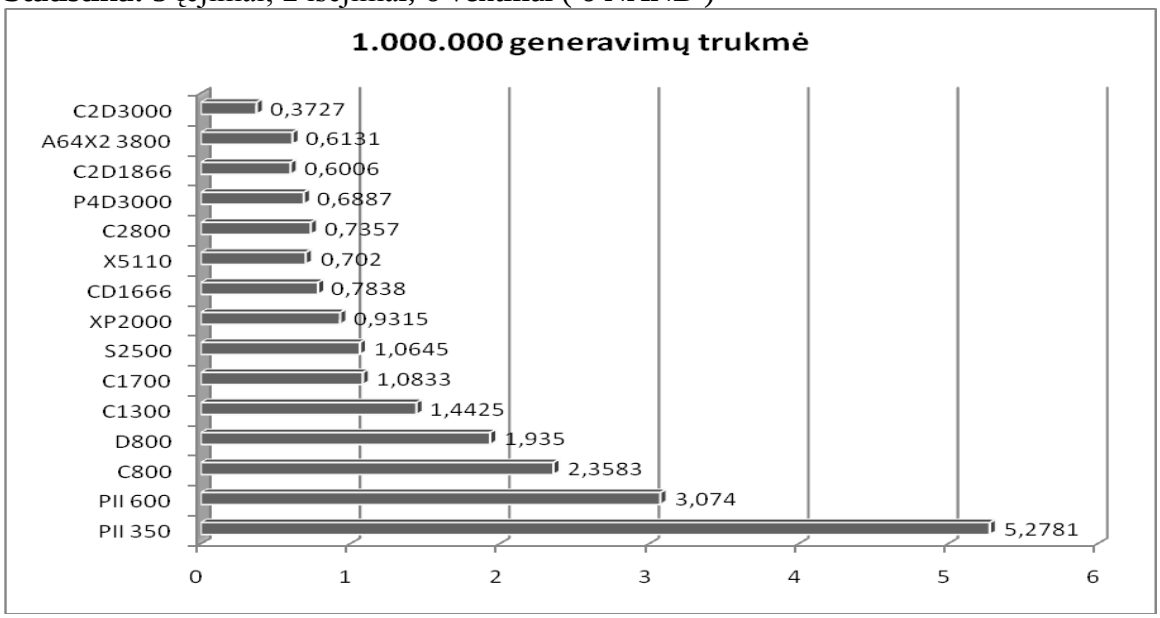
99 pav. b08 300.000 generavimų greitaveika



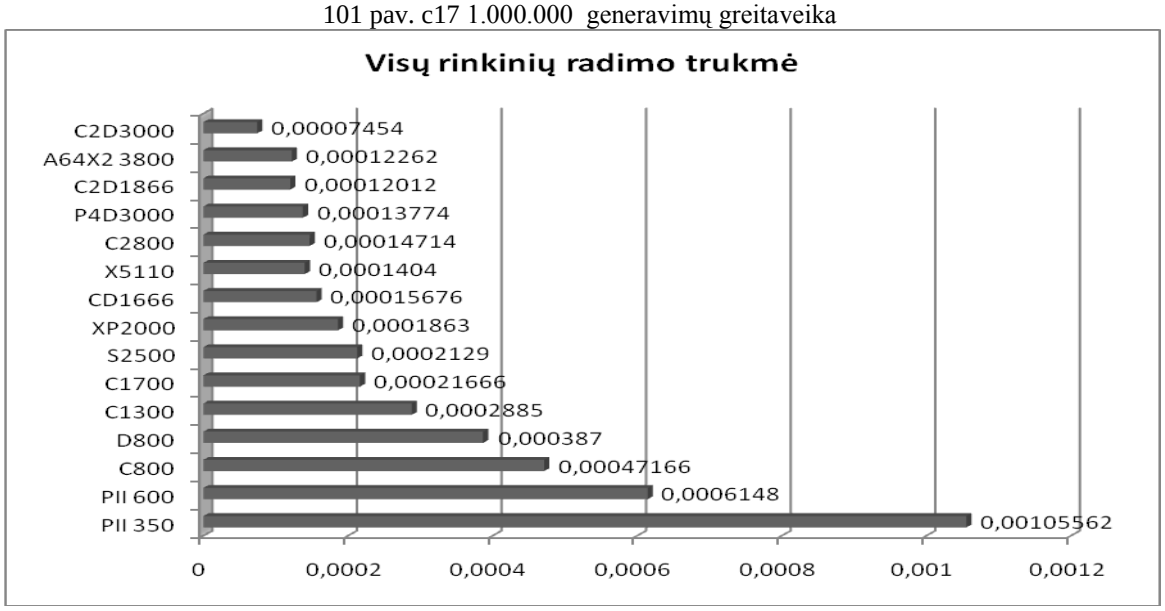
100 pav. b08 rastų K narių priklausomybė nuo generavimų skaičiaus

ISCAS85 c17

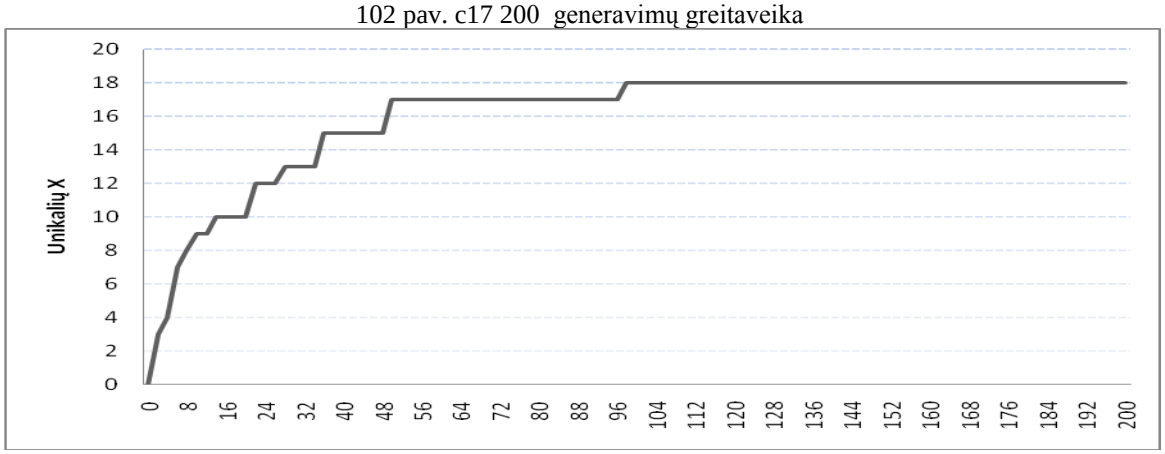
Statistika: 5 įėjimai; 2 išėjimai; 6 ventiliai (6 NAND)



101 pav. c17 1.000.000 generavimų greitaveika



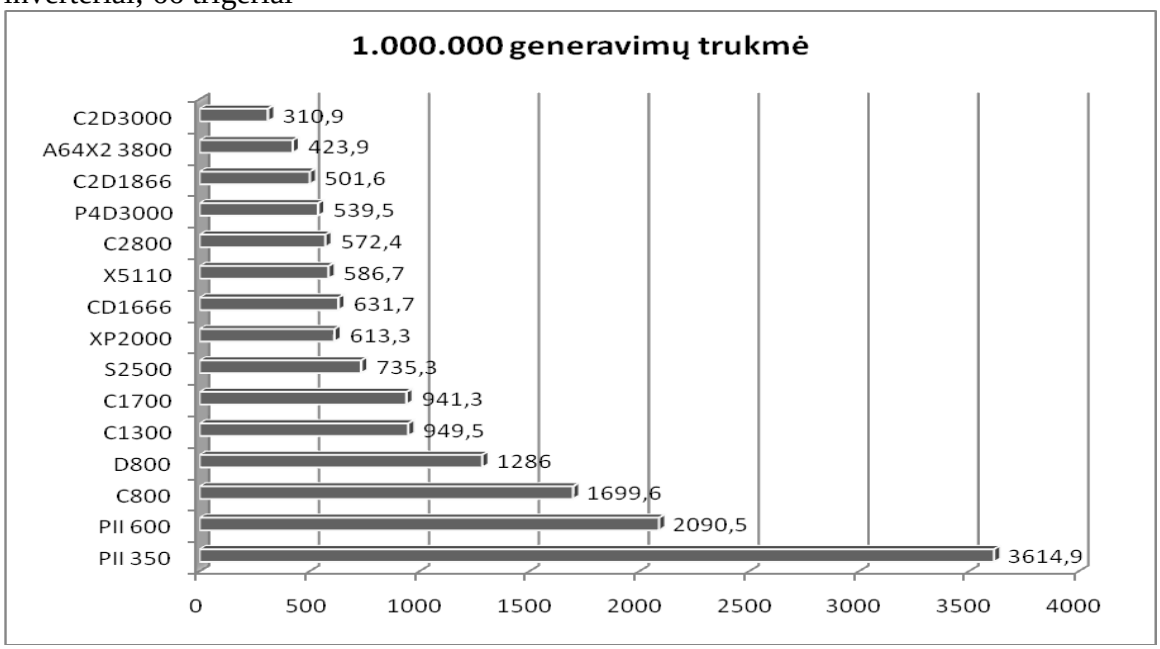
102 pav. c17 200 generavimų greitaveika



103 pav. c17 rastų K narių priklausomybė nuo generavimų skaičiaus

ISCAS99 b04

Statistika: 11 įėjimų; 8 išėjimai; 632 ventiliai (35 AND + 482 NAND + 30 OR); 105 inverteriai; 66 trigeriai



104 pav. b04 1.000.000 generavimų greitaveika