ŠIAULIŲ UNIVERSITETAS
TECHNOLOGIJOS FAKULTETAS
ELEKTRONIKOS KATEDRA

TOMAS VISMANTAS

JTAG SĄSAJA PROGRAMUOJAMUOSE ELEKTRONINIUOSE PRIETAISUOSE

Magistro darbas

Signalų technologija, elektronikos inžinerija (RM 03)

Mokslinis vadovas

Doc.dr: G. Daunys

Šiauliai, 2005

SANTRAUKA

Tomas Vismantas

JTAG SĄSAJA PROGRAMUOJAMUOSE ELEKTRONINIUOSE PRIETAISUOSE

Magistro darbas

Magistro darbe aprašytas JTAG (boundary scan) interfeisas. Jame aptariamas IEEE standarto 1149.1 grandinės modelis bei TAP (Test Access Port) kontrolerio pagrindinės instrukcijos. Atlikta programuojamų loginių integrinių mikroschemų apžvalga: evoliucija, pirmaujančių gamintojų (ALTERA, XILINX, ACTEL) produkcijos ir programinės įrangos įvertinimas. Pateikti patarimai, kaip reikia išsirinkti tinkamą programuojamą loginį įrenginį. Išsamiai aprašomos tiriamosios mikroschemų šeimos XC9500 charakteristikos, ypatybės ir privalumai. Bendrais bruožais apibūdinta programavimo kalbos VHDL reikšmė. Pateikti keli detalūs projekto kūrimo aprašymai, naudojant schemos braižymo ir VHDL kalbos metodus. Patvirtinama autoriaus suformuluota mokslinio darbo hipotezė, kad naudodami savo projektuose JTAG sąsają turinčius programuojamus loginius prietaisus galėsime sutaupyti laiko, ploto ir pagerinsime jų kokybę. Tai yra ateities technologija, kuri greitai bus naudojama ir Lietuvoje.

SUMMARY

Tomas Vismantas

JTAG interface of programmable electronic devices

Master‘s work

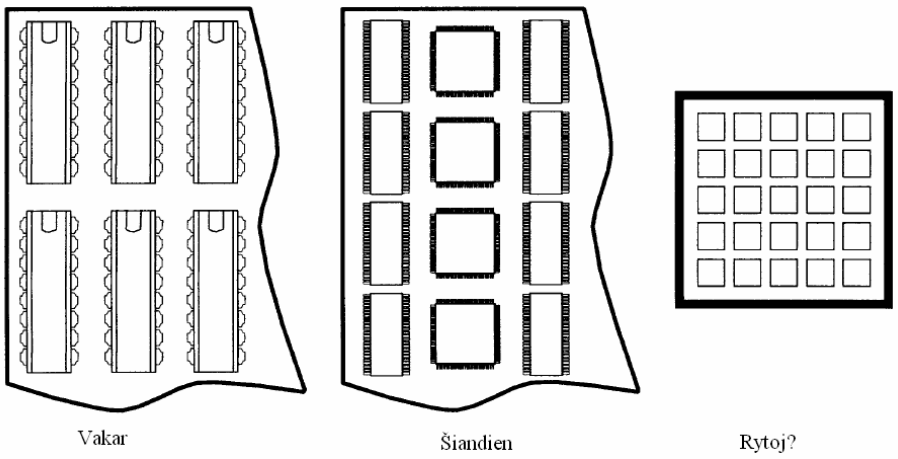
This master‘s final paper describes JTAG (boundary scan) interface in which discuss IEEE standart 1149.1 circuit model and the main TAP (Test Access Port) controllers instructions. Accomplished programmable integral logical ICs overview: development, leading manufacturer (ALTERA, XILINX, ACTEL) production and programmable equipment evaluation. Represented recomendation, how we can pick suitable programmable logical device. The paper presents detailed describe searching ICs family XC9500 characteristic, features and merits. In general terms presented programmable logic language VHDL value. It also produces some detailed compose describes of the project, using methods of circuit drawing and VHDL language. Master‘s hypothesis that if we will use JTAG interface processed logical programmable instrumentation in our projects we can save up time, area and improve their quality is confirmed. This is prospective technology which also soon will be in use in Lithuania.

TURINYS

1. ĮVADAS

Temos aktualumas. Praeityje daug dėmesio buvo skiriama *boundary scan* teikiamai naudai bandymų procesams ir bandymų inžinerijai. Kol galų gale diskusija į *boundary scan* naudojimą duotame projekte bus paremta pozityvia produkto įtaka ilgaamžiškumo vertei, didėjanti projekto nauda dažnai yra nepastebima. Šiame darbe yra aprašoma projekto nauda visiems projektavimo lygiams: lustui, montažinei plokštei ir sistemai.

1985 pradžioje keletas Europos ir Šiaurės Amerikos kompanijų susivienijo ir įkūrė Joint Actions Group (JTAG). Pirmoji jų užduotis buvo išspręsti spausdintų montažo plokščių gamybos testavimo problemą, kuri vis labiau aštrėjo, nes integrinės schemos (ICs) tapo mažesnės ir kompleksiškesnės (žr. pav.1 ir pav.2)[1,2]. Galiausiai jų sprendimas buvo standartizuotas kaip IEEE Std 1149.1-1990 Test Access Port and Boundary-Scan struktūra. Standartas numato, reikiami testavimo resursai turi būti pačiose integrinėse schemose[3].

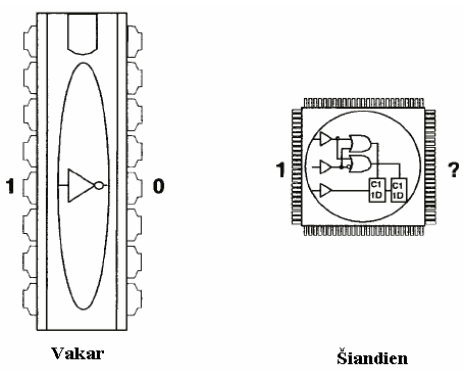


pav. 1-1 nuostabūs spausdinto montažo plokščių ploto sumažėjimo rezultatai, prarandant priėjimo galimybę testuoti

Pramoninis spausdinto montažo plokščių testavimas yra būtinas bandant surasti defektus (tokius kaip trumpas jungimas tarp kontaktų arba šaltas litavimas) montuojant integrines schemas ir kitus komponentus ant plokštės. Kuo integrinės schemos bei kitos dalys yra mažesnės ir smulkesnės, tuo yra sunkiau tai padaryti. Testo funkcionavimo galimybė (kraštinė jungtis) išskirti spausdinto montažo plokštės atitinkamo lygio surinkimo defektus greičiau prieštarauja didėjančiam

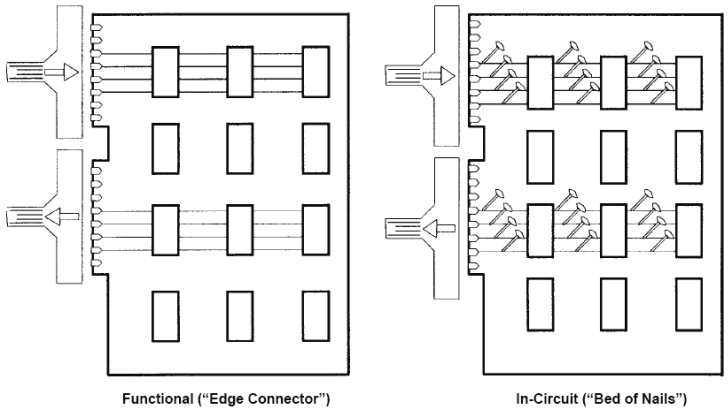
[1] Harry Bleeker, Peter van den Eijnden, Frans de Jong, Boundary-Scan Test - A Practical Approach, Kluwer Academic Publishers, 101 Philip Drive, Assinippi Park, Norwell, MA 02061, 1993.
[2] Colin M. Maunder, Rodham E. Tulloss, ed., The Test Access Port and Boundary‘-Scan Architecture, IEEE Computer Society Press, 10662 Los Vaqueros Circle, PO Box 3014, Los Alamitos, CA 90720-1264, 1990.
[3] IEEE Std 1149.1-1990 (includes IEEE Std 1149.1a-1993), IEEE Standard Test Access Port and Boundary-Scan Architecture, Institute of Electrical and Electronics Engineers, 345 East 47th Street, New York, NY 10017, October 1993.

funkcionalumo tankiui (žr. pav.3). Iki tol kol pradiniai įėjimai\išėjimai yra naudojami testavimui, testavimo kartos sudėtingumas ir reikiamas testavimo laikas eksponentiškai augs didėjant montažinių plokščių kompleksiškumui.



pav. 1-2 didėjant lusto integracijos laipsniui komplikuojasi kontroliavimas

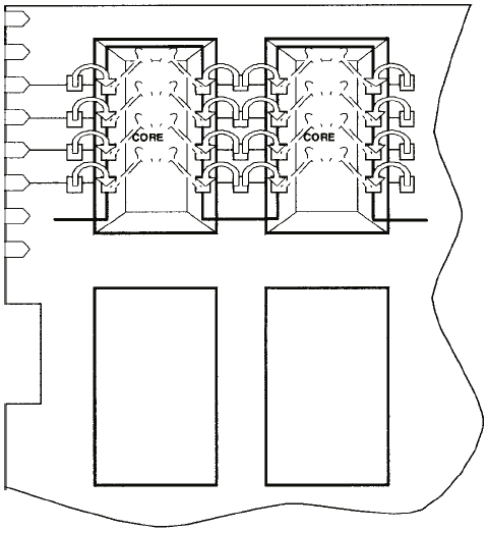
Testo schema (testavimo salelės), kuri buvo naudojama ankščiau norint aptikti defektą sudėtingose plokštėse, taip pat prieštarauja augančiam integrinių schemų funkcionalumui ir be to pagal fizinius parametrus (žr. pav.3). Nuo tada kai testavimo technologija yra paremta fiziniu testavimu kai testuojami (geriau visi) vidiniai sujungimai ant montažinės plokštės, maži tarpai tarp kojelių reikalauja tobulesnių testerių, kurie yra sudėtingesni ir brangesni. Daugeliu atvejų, tokiais kaip multilustiniai moduliai, daugiasluoksnis montažas ir persidengę signalų takeliai, fizinis prisijungimas prie takelių yra fiziškai neįmanomas. Didėjant integrinių schemų integracijos laipsniui atsiranda problema, nes norint lustų išėjimus nustatyti į žinomus loginius lygius testavimui, įrenginio įėjimo funkcija turi būti manipuliuojama ilga ir sudėtinga kombinacijų seka. Toks pat argumentas taikomas ir nepertraukiamumo testui lusto įėjimuose. [4,5,6]



pav. 1-3 Tradiciniai spausdinto montažo plokštės testavimo metodai

[4] Harry Bleeker, Peter van den Eijnden, Frans de Jong, Boundary-Scan Test - A Practical Approach, Kluwer Academic Publishers, 101 Philip Drive, Assinippi Park, Norwell, MA 02061, 1993.
[5] M. Maunder, Rodham E. Tulloss, ed., The Test Access Port and Boundary'-Scan Architecture, IEEE Computer Society Press, 10662 Los Vaqueros Circle, PO Box 3014, Los Alamitos, CA 90720-1264, 1990.
[6] Kenneth P. Parker, The Boundary-Scan Handbook, Kluwer Academic Publishers, 101 Philip Drive, Assinippi Park, Norwell, MA 02061, 1992.

Boundary scan idėja remiasi testavimo technologijos principu. Kad fiziniai testeriai, kurie yra patalpinami laido viduryje yra pakeičiami boundary scan celėmis. Šie virtualūs testeriai yra patalpinami lusto įėjimuose ir išėjimuose ir dėl to jie atsiranda kontakto gale (žr. pav.4). Tai įtakoja du labai svarbius patobulinimus: 1)fizinis priėjimas prie boundary scan kontakto daugiau nėra reikalingas, ir 2) nepertraukiamumo testas nėra priklausomas nuo lusto integracijos laipsnio. Tinklo efektas yra tas, kad pramoninio testavimo tikslas yra surasti defektus tarp kojelės arba jungiamojo laido, gali būti užbaigtas automatizuota naujos kartos testavimo technologija (automated test-pattern generation (ATPG))[7].



pav. 1-4 Boundary-Scan idėja

Galų gale, kad boundary scan celės su minimaliu kojelių perkaitinimu galėtų aprūpinti serijine gamyba, savarankiška kontrole ir priežiūra, jos yra suprojektuotos taip, kad galėtų nuosekliai apjungti nuo perstūmimo registro iki dviejų lusto kojelių tarpo, Test Data Input (TDI) ir Test Data Output (TDO). Papildomos valdymo struktūros, kurios reikalauja pasirinkti tarp normalaus ir testavimo darbo režimo, buvo suprojektuotos siekiant minimizuoti lusto kojelių perkaitinimą ir maksimizuoti testo režimo lankstumą bei panaudoti spausdinto montažo plokščių pramoninį testavimą.

Tyrimo objektas. JTAG interfeiso taikymo galimybės.

Tyrimo metodas. Formuluojant tyrimo hipotezę, tikslus ir uždavinius bei siekiant išryškinti JTAG sąsajos privalumus buvo pasitelkta techninės dokumentacijos analizė ir praktinis principinės schemos braižymo ir VHDL programavimo metodas.

Tyrimo tikslas - JTAG panaudojimas programuojamuose elektroniniuose prietaisuose (studentų mokymui).

[7] Colin M. Maunder, Rodham E. Tulloss, ed., The Test Access Port and Boundary'-Scan Architecture, IEEE Computer Society Press, 10662 Los Vaqueros Circle, PO Box 3014, Los Alamitos, CA 90720-1264, 1990.

***Tyrimo uždaviniai**:*

1. Suprojektuoti JTAG interfeiso sąsajos montažinę plokštę.
2. Išsiaiškinti JTAG veikimo principus ir privalumus, elektroniniuose prietaisuose.
3. Ištirti programuojamos logikos veikimą panaudojant XC95108 mikroschemą.

Mokslinis naujumas. JTAG yra sparčiai plintanti technologija technologiškai išsivysčiusiose šalyse. Tačiau Lietuvoje, kiek yra žinoma, nėra pasirodęs nė vienas darbas keliantis šį tikslą. Tai ir sudaro tyrimo originalumą ir mokslinį naujumą. Tyrimas atveria galimybę ne tik įvertinti JTAG interfeiso taikymo galimybes, bet ir išryškinti teorinę ir praktinę jo svarbą.

Tyrimo rezultatų aprobavimas. Magistrinio tyrimo rezultatai pristatyti konferencijoje:

2005. 05. 25 Studentų mokslinė konferencija. Pranešimo tema: JTAG sąsaja programuojamuose elektroniniuose prietaisuose.

***Tyrimo struktūra**:* Magistro darbą sudaro įvadas, penki skyriai, išvados, literatūros sąrašas. Darbo apimtis

Įvade aprašomas temos aktualumas, tyrimo objektas, metodas, tikslas, uždaviniai, mokslinis naujumas, tyrimo rezultatų aprobavimas.

Pirmame skyriuje kalbama apie JTAG (boundary scan) interfeisą, kuriame aptariamas IEEE standarto 1149.1 grandinės modelis bei TAP (Test Access Port) kontrolerio pagrindinės instrukcijos.

Antrame skyriuje atliekama programuojamų loginių integrinių mikroschemų apžvalga: evoliucija, pirmaujančių gamintojų (ALTERA, XILINX, ACTEL) produkcijos ir programinės įrangos įvertinimas ir aptarimas. Patariama kaip reikia išsirinkti tinkamą programuojamą loginį įrenginį.

Trečiame skyriuje aprašomos tiriamosios mikroschemų šeimos XC9500 charakteristikos, ypatybės ir privalumai.

Ketvirtame skyriuje aptariamos VHDL programavimo kalbos reikšmė.

Penktame pateikiamas JTAG praktinis taikymas, t.y. principinės schemos ir VHDL programavimo kalbos metodai.

Paskutiniame skyriuje pateikiamos darbo išvados.

2. JTAG (BOUNDARY SCAN) INTERFEISAS

JTAG yra vidinis interfeisas naudojamas testavimo, derinimo ir programavimo užduotims atlikti.

IEEE 1149.1 Boundary Scan Architecture (tas pats JTAG) standartas sukurtas sudėtingų loginių schemų testavimui, montažinėje plokštėje. Šitokiu būtu gali būti testuojami šiuolaikiniai procesoriai, funkciniai sisteminių plokščių mazgai, praplėtimo plokštės (PCI magistralė taip pat turi JTAG signalų interfeisą).

2.1 IEEE STD. 1149.1 BST ARCHITEKTŪRA

Įrenginiai palaikantys IEEE Std. 1149.1 BST rėžimą naudoja keturis TDI, TDO, TMS, ir TCK pagrindinius signalus, ir dar vieną papildomą TRST signalą. Lentelėje 2.1-1 pateiktas signalų funkcijų aprašymas.

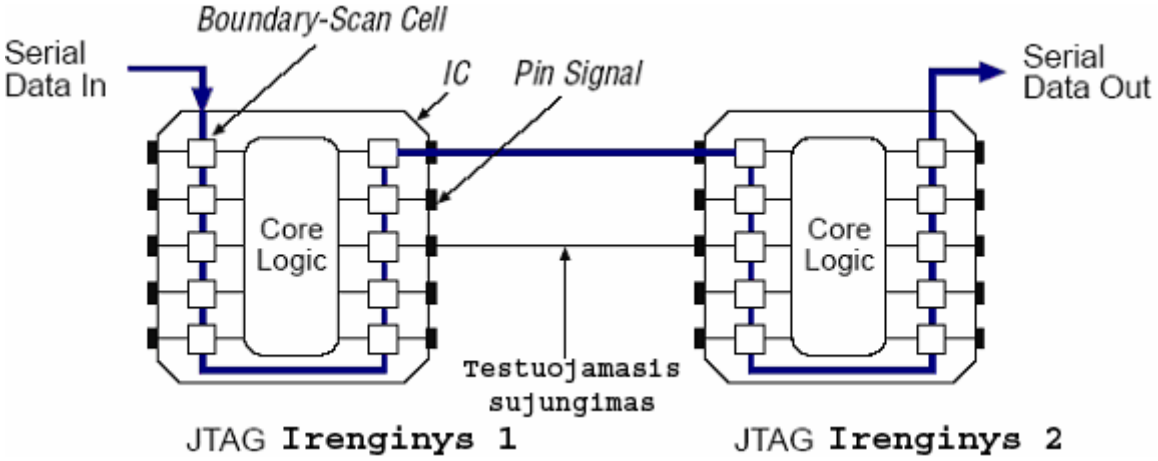
Prievadas	Aprašymas	Funkcijos
TDI (Test Data Input)	Testuojamų duomenų įėjimas	Nuoseklių duomenų įėjimo prievadas skirtas instrukcijų, testavimo bei programavimo duomenims siųsti. Duomenys yra perduodami ties augančiu TCK frontu.
TDO (Test Data Output)	Testuojamų duomenų išėjimas	Nuoseklių duomenų išėjimo prievadas skirtas instrukcijų, testavimo bei programavimo duomenims siųsti. Duomenys yra perduodami ties krintančiu TCK frontu. Prievadas yra trijų būsenų, jeigu duomenys nepradėti perdavinėti į įrenginio išėjimą, prievadas yra trečioje būsenoje.
TMS (Test Mode Select)	Testo rėžimo išrinkimas	Įėjimo prievadas, kuris formuoja kontrolinius JTAG signalus. Test Access Port TAP controler duomenys yra perduodami ties augančiu TCK frontu. Taigi, TMS turi būti nustatomas prieš kylantį TCK frontą. TMS yra įvertinamas ties kylančiu TCK frontu.
TCK (Test Clock)	Testavimo sinchronizacijos įėjimas	Sinchronizavimo signalo įėjimo prievadas boundary-scan test vienos komandos vykdomos ties kylančiu TCK frontu, o kitos ties krintančiu TCK frontu.
TRST	Testo atstatymo (reset) įėjimo prievadas	Aktyvus žemo lygio įėjimas kuris asinchroniškai atstato (reset) boundary-scan grandinę. Tai neprivalomas pagal IEEE Std. 1149.1 standartą prievadas.

lentelė 2-1 IEEE Std. 1149.1 prievadų aprašymas

Šie keturi TDI, TDO, TMS, ir TCK pagrindiniai signalai sudaro *TAP* (Test Access Port) interfeisą, per kurį testuojamoji aparatūra jungiama prie testuojančiosios. Testuojančioji aparatūra

formuoja testinius signalus pagal testavimo programą, kurią pateikia testuojamojo įrenginio gamintojas, o gauti testo duomenys, testavimo pabaigoje, palyginami su gamintojo etalonu.

Jei JTAG grandyje yra keletas testuojamų įrenginių, tai į kiekvieną įrenginį galima kreiptis individualiai pagal jo IDCODE. Bet kokios skaitmeninės schemos palaikančios JTAG interfeisą testavimo principas parodytas pav. 2-1

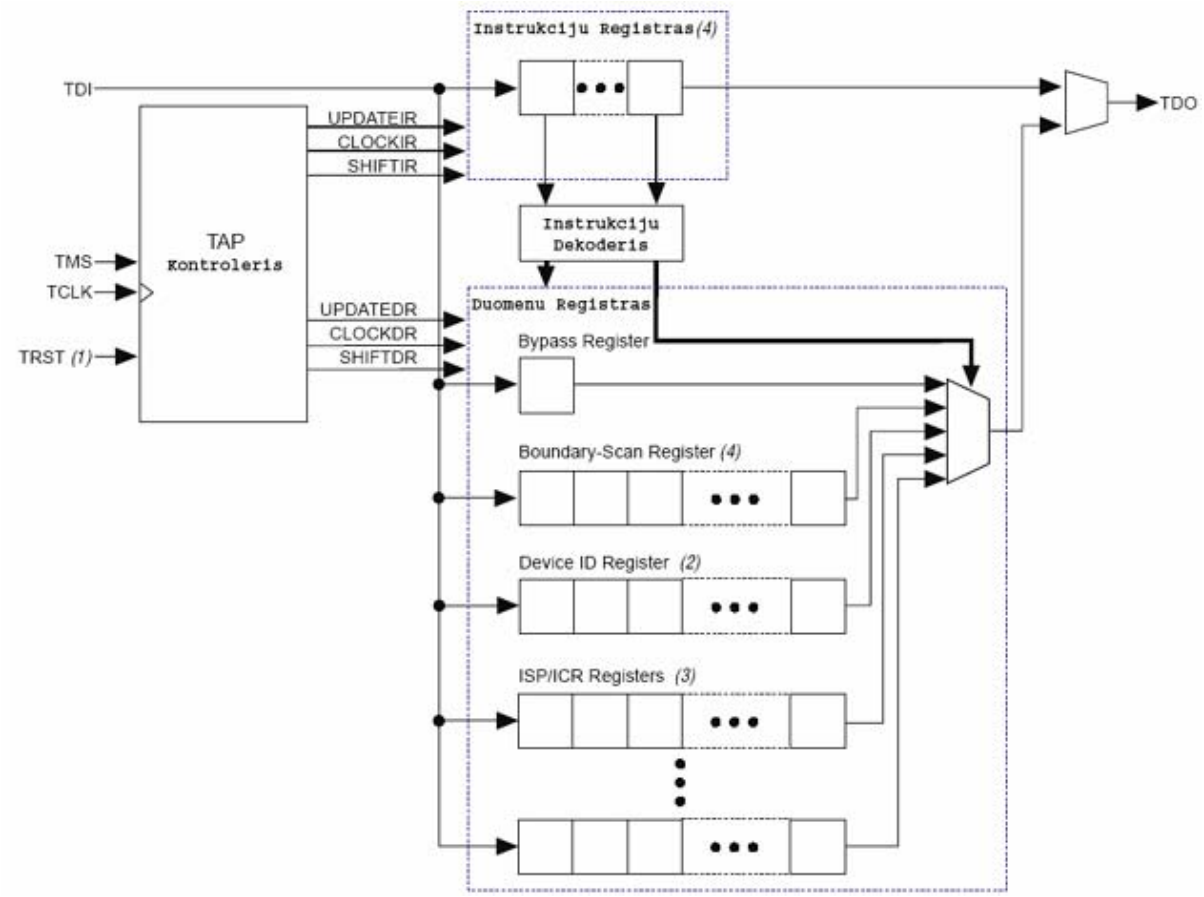


Pav. 2-1 IEEE Std. 1149.1 Boundary-Scan Testavimas

Paveikslėlyje pavaizduotoje skaitmeninėje schemoje yra įėjimo, išėjimo (gali būti trijų būsenų) ir dvikrypčiai signalai. Testavimo makroląstelės (Boundary-Scan Cell) yra tarp realių įrenginio išvadų ir vidinės logikos — tai yra jos išdėstytos palei įrenginio kraštą (boundary). TAP kontroleris gali skanuoti, valdyti ir nuskaityti informacija iš makroląstelės. Iš čia ir kilęs pavadinimas Boundary Scan, kurį galima išversti kaip „krašto skanavimas“. Įjungus testavimo režimą TAP kontroleris sugeba logiškai atjungti signalus nuo išorinių išvadų, po to pasiųsti savo testavimo duomenis į įėjimų vidinius išvadus ir nuskaityti gautus rezultatus. Tai viskas ko reikia testuojant nuoseklias schemas (įrenginius su atmintim). JTAG interfeisas yra patrauklus tuo, kad nepriklausomai nuo aparatūros sudėtingumo ji testuojama tik per keturis signalinius laidus.

Testinė logika naudojama įrenginiuose palaikančiuose JTAG, susideda iš šių elementų: pav. 2-2

1. Testinio TAP prievado (keturių jungčių).
2. TAP kontrolerio valdančio duomenų registrą.
3. Instrukcijų registro IR (Instruction Register), kuris priima nuoseklų kodą iš TDI įėjimo. Instrukcijos kodas reikalingas testavimo funkcijos arba duomenų registro į kurį kreipiamės išrinkimui.
4. Testinių duomenų registrų: Bypass Register, Boundary Scan Register ir Device Identification Register.



Pav. 2-2 funkcinis IEEE Std. 1149.1 standarto grandinės modelis

(1) neprivalomas pagal IEEE Std. 1149.1 standartą prievadas.

(2) įrenginio identifikavimo registras ID yra visuose su JTAG suderinamuose įrenginiuose.

(3) privatus registrai yra naudojami programuojamumui sistemoje palaikyti ir konfigūracijos keitimui grandinėje.

(4) priklausomai nuo įrenginio, šio registro ilgis gali būti skirtingas.

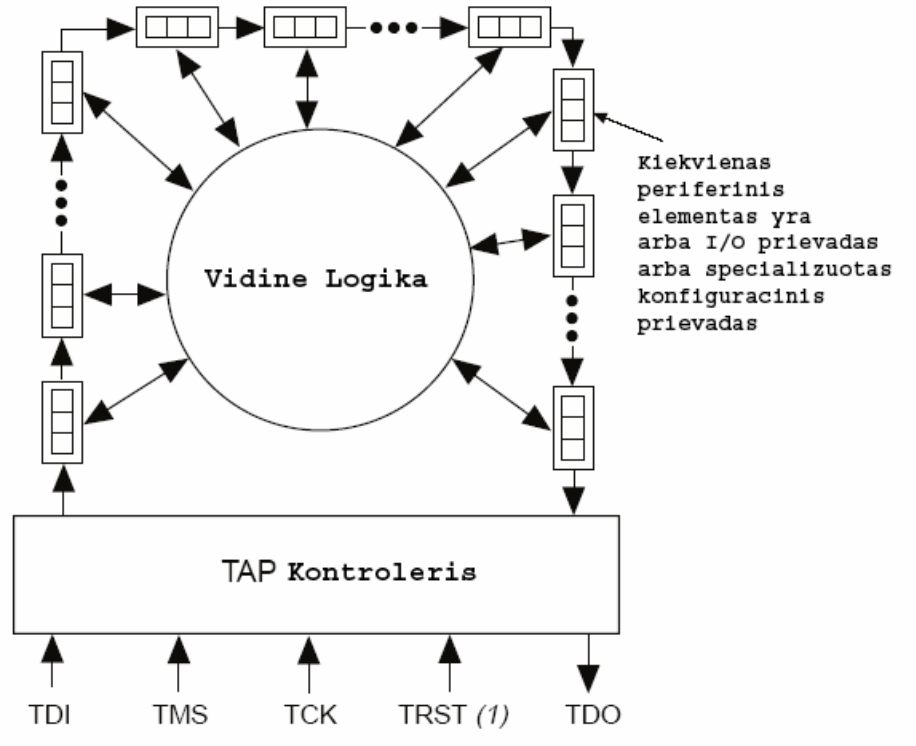
2.2 IEEE STD. 1149.1 BST STANDARTO GRANDINĖS REGISTRAI

Instrukcijų ir ***duomenų*** registrai yra tarpusavyje nepriklausomi perstūmimo registrai, kurie sujungti lygiagrečiai. Į šių registrų įėjimų (vyresniuosius bitus) patenka TDI signalas, o iš išėjimų (jaunesnieji bitai) formuojamas TDO signalas. Per kiekvieną teigiamą poslinkį duomenys yra perstumiami per vieną bitą.

Bypass registras yra vieno bito. Jis naudojamas kaip trumpiausias nuoseklių duomenų apėjimo kelias, kai kiti registrai nenaudojami.

Boundary Scan registras yra ilgas perstūmimo registras, kuris susideda iš visų įrenginio makroląstelių. Dvikrypčiams signalams (arba jų grupėms), be ***Boundary Scan*** registro informacinių makroląstelių, atitinkamų išorinių signalų makroląstelių, yra ir valdančiosios makroląstelės, išrenkančios informaciją nešančių makroląstelių darbo rėžimą.

Boundary-scan registrą galima panaudoti išorinių prievadų kontaktams testuoti arba vidinių duomenų nuskaitymui. pav. 2-3 parodyta kaip testuojami duomenys yra nuosekliai ratu perstumiami per visą IEEE Std. 1149.1 standarto įrenginio periferiją.



Pav. 2-3 Boundary-Scan registras

(1) TRST tai neprivalomas pagal IEEE Std. 1149.1 standartą prievadas.

32 bitų ***Device Identification*** registras yra įrenginio identifikavimo registras sudarytas iš įrenginio kodo ir versijos numerių, kuriuos suteikia gamintojas. Iš ***Device Identification*** registro TAP kontroleris sužino, su kuriuo įrenginiu vyksta duomenų mainai.

Be šių privalomų registrų, įrenginys gali turėti specializuotus papildomus registrus. Pavyzdžiui, procesoriai su JTAG interfeisu turi vieno bito RUNBIST registrą iš kurio galima nuskaityti BIST (Built-in Self Test) testo rezultatus. Prieš BIST paleidimą į šį registrą įrašomas vienetas. Jei baigus vykdyti testą registro reikšmė pasikeičia į nulį tai reiškia, kad testas praėjo sėkmingai.

4 bitų ***Instrukcijų registras*** leidžia išrinkti testą ir adresuoti jį vienam iš prieš tai paminėtų registrų. Iš 16-likos galimų instrukcijų kodų visiems įrenginiams veikiantiems pagal IEEE Std. 1149.1 standartą privalomi yra šie instrukcijų kodai: EXTEST, SAMPLE, IDCODE ir BYPASS

Kiti instrukcijų kodai gali būti nustatomi konkretiems testuojamiesiems įrenginiams. Pavyzdžiui, procesoriai turi RUNBIST instrukciją pagal kurią aktyvuojamas BIST testas.

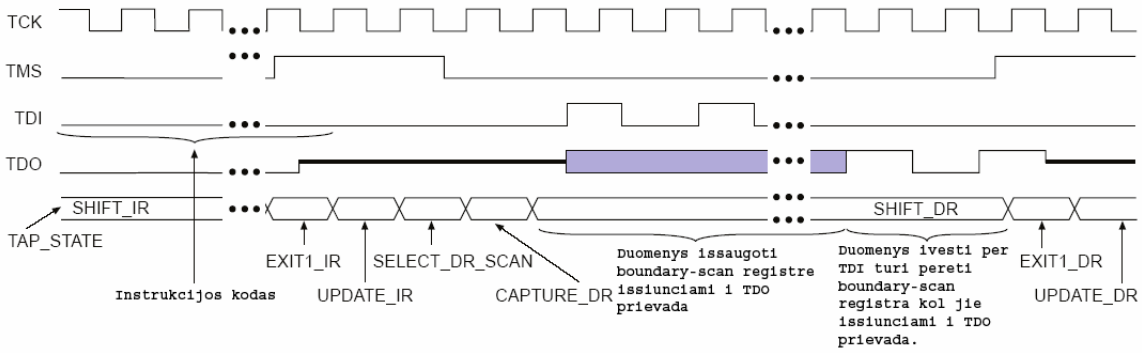
2.3 IEEE STD.1149.1 STD. PRIVALOMOS INSTRUKCIJOS

2.3.1 EXTEST INSTRUKCIJOS REŽIMAS

Instrukcija **EXTEST** (kodas 0000) skirta išoriniams įrenginio sujungimams testuoti. Testo metu į išėjimo linijas pasiunčiami signalai prieš tai įrašyti į **Boundary-Scan** registrą, o išėjimų signalai nuskaitomi į šį registrą. Dvikrypčiai signalai konfigūruojami atitinkamai pagal kryptį nustatančias makroląsteles **Boundary-Scan** registre.

2.3.2 SAMPLE/PRELOAD INSTRUKCIJOS REŽIMAS

Instrukcija **SAMPLE/PRELOAD** (kodas 0001) turi dvi funkcijas. Kai TAP kontroleris yra Capture-DR būsenoje, ši instrukcija leidžia padaryti visų išorinių signalų momentinę „nuotrauka“ nepertraukiant normalaus įrenginio darbo rėžimo. Signalų reikšmės yra fiksuojamuos ties kylančiu TCK frontu. Update-DR būsenoje ši instrukcija įkrauna duomenis į išėjimų makroląsteles (bet ne į išėjimus), iš kur nuosekliai bus išsiunčiami į išorinius išėjimus panaudojant EXTEST instrukciją. Duomenys iš anksto įkraunami ties krintančiu TCK frontu pav. 2-4



Pav. 2-4

2.3.3 IDCODE INSTRUKCIJOS REŽIMAS

Instrukcija **IDCODE** (kodas 0010) prie interfeiso prijungia ***Device Identification*** registrą ir leidžia iš jo nuskaityti duomenis (***Device Identification*** registras yra tik skaitymui).

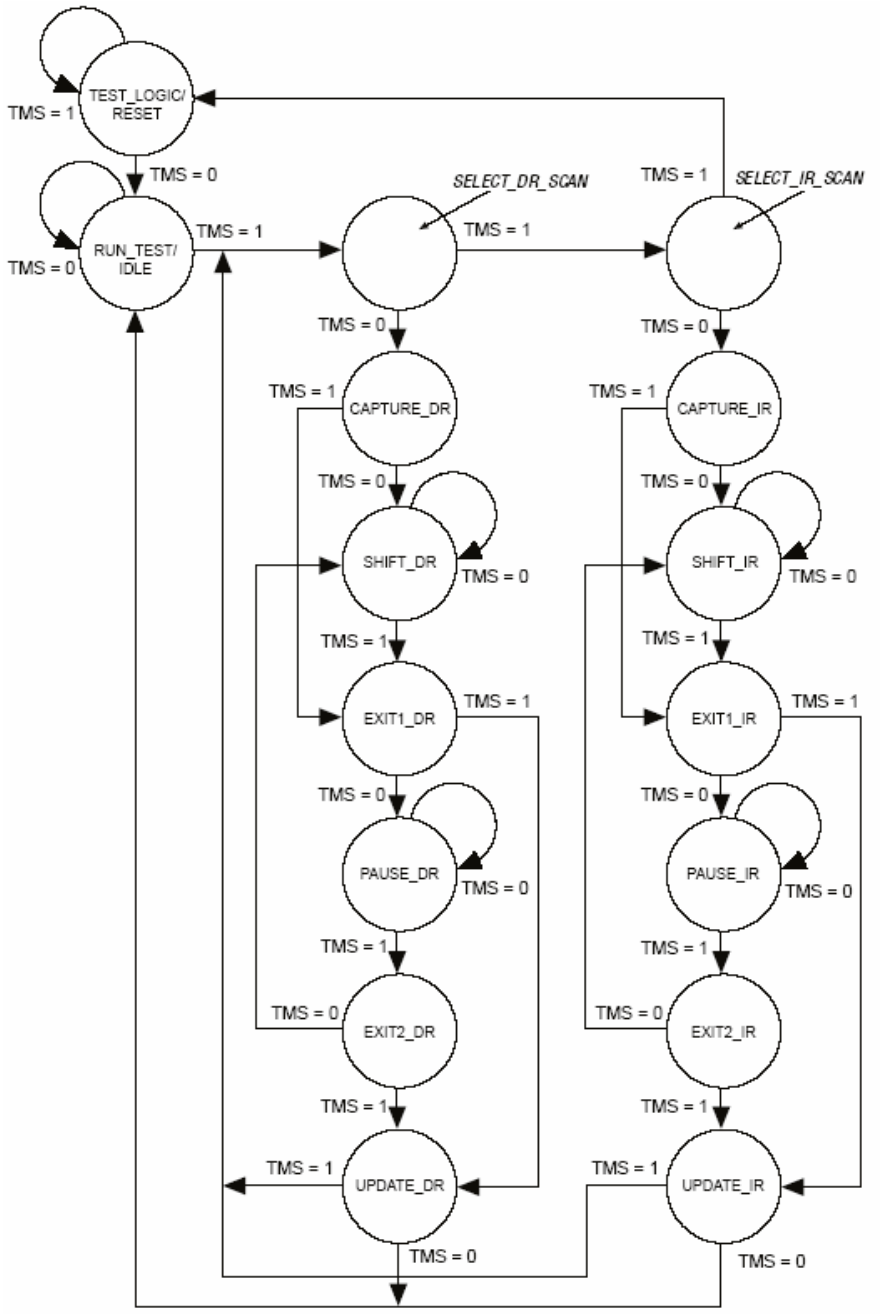
2.3.4 BYPASS INSTRUKCIJOS REŽIMAS

Instrukcija **BYPASS** (kodas 1111) skirta vieno bito apėjimo registrui prie interfeiso prijungti. Šis registras tiesiog praleidžia duomenis trumpiausiu keliu per įrenginį ir nereaguoja į duomenų srautą. TDI įėjimas paprastai būna prijungiamas prie aukšto lygio „pritraukiančiojo“ rezistoriaus, todėl nutrūkus JTAG grandinei, visi įrenginiai nuo nutrūkimo vietos bus prijungiami prie apėjimo registrų. Tai prevencija prieš neprognozuojamą įrenginio veikimą grandinės nutrūkimo atveju.

2.4 IEEE STD. 1149.1 TEST ACCESS PORT (TAP) KONTROLERIS

IEEE Std. 1149.1 test access port (TAP) kontroleris yra 16-kos būsenų automatas. Jo būsena yra fiksuojama ties kylančiu TCK frontu, panaudojant TMS prievadą įrenginio valdymui pagal IEEE Std. 1149.1 standartą. pav. 2-5 pavaizduotas TAP kontrolerio būsenų mechanizmas.

1. Į pradinę ***Test-Logic-Reset*** būseną kontroleris automatiškai patenka po maitinimo įtampos įjungimo momento arba gali būti priverstinai nustatomas į šią būsena, iš bet kurios kitos būsenos, TMS signalą nustatant į „1“ ne mažiau kaip penkiems TCK ciklams. Šioje būsenoje logikos testavimas negalimas o įrenginys dirba normaliame režime.
2. Būsena ***Run-Test/Idle*** yra tarpinė būsena tarp testinių operacijų vykdymo. Šioje būsenoje registrai nekeičia savo reikšmių.
3. ***Capture-DR*** būsenoje vykdant instrukcijas EXTEST ir SAMPLE/ PRELOAD, skanuojantysis registras fiksuoja duomenis tik įėjimo linijose.
4. ***Shift-DR*** būsenoje duomenys iš TDI perstumiami per prijungtą perstūmimo registrą į TDO.
5. Būsenoje ***Pause-DR*** kontroleris laikinai uždraudžia duomenų judėjimą per perstūmimo registrą.
6. ***Update-DR*** būsenoje ties krentančiu TCK frontu signalai iš perstūmimo registro fiksuojami testinės išėjimo makroląstelės.
7. Būsenoje ***Capture-IR*** kontroleris į perstūmimo registrą užkrauna instrukcijos kodą 0001 SAMPLE — tai neutrali instrukcija.
8. Būsenoje ***Shift-IR*** į grandinę tarp TDI ir TDO įterpiamas instrukcijų perstūmimo registras, bet tuo momentu dar vykdoma prieš tai buvusi instrukcija.
9. Būsenoje ***Pause-IR*** kontroleris laikinai uždraudžia duomenų judėjimą per instrukcijų perstūmimo registrą.
10. ***Update-IR*** būsenoje ties krentančiu TCK frontu fiksuojama naujai vykdoma instrukcija ir į TDI — TDO grandinę įjungiamas atitinkamas registras.



Pav. 2-5

Be šių pagrindinių kontrolerio būsenų, aprašančių testuojamojo įrenginio veikimą, yra ir laikinų tarpinių būsenų, kurios būtinos automato perėjimams realizuoti. Joms priskiriamos ***Select-DR-Scan, Exitl-DR, Exit2-DR, Select-DR-Scan, Exitl-IR*** ir ***Exit2-IR***. Valdančio automato būsenų ir perėjimų schema pavaizduot pav. 2-5 Prie būsenos perėjimo rodyklių nurodytos TMS signalo reikšmės TCK fronto momentu.

JTAG interfeisas turi savo specializuotą aprašymo kalbą BSDL (Boundary Scan Description Language) kuri savo ruožtu yra standartinės aparatūros aprašymo kalbos (VHSIC Hardware Description Language - VHDL) poaibis. Informacinių ir valdančiųjų makroląstelių sudėtis ir

eiliškumas kiekvieno įrenginio duomenų perstūmimo registre yra specifiniai ir yra skelbiami gamintojų (tam ir yra reikalingas 32 bitų ***Device Identification*** registras).

Pentium procesoriuose TAP prievado panaudojimas yra patobulintas: įvedant papildomą R/S pertraukimų signalą, pagal kuri, procesorius pereina į *zondavimo/ derinimo* rėžimą. Šiame režime naudojant papildomas TAP instrukcijas įmanomas „bendravimas“ su procesoriaus registrais. Tokiu būdu gali būti realizuojamos derinimo priemonės, visiškai nepriklausančios (ir neblokuojamos) nuo procesoriaus vykdomo programos kodo.

3. PROGRAMUOJAMŲJŲ LOGINIŲ INTEGRINIŲ SCHEMŲ APŽVALGA

Prieš keletą metų paaiškėjo, kad programuojamuose loginėse integralinėse schemose (PLIS) — patogi įsisavinimui ir panaudojimui elementinė bazė, kuri neturi analogų. Pastaraisiais metais labai padidėjo elementų, patalpinamų į kristalo ploto vienetą, skaičius. Dauguma pirmaujančiųjų gamintojų arba pradėjo serijinę gamybą, arba paskelbė apie PLIS kurių ekvivalentinė loginė talpa yra daugiau kaip 1 mln. loginių ventilių. O PLIS kainos per metus sumažėja apie 10 kartų ir ši tendencija kol kas išlieka ir toliau.

Tokia situacija rinkoje sukėlė daug klausimų, susijusių su specialistų ruošimu kurie sugebėtų projektuoti skaitmeninių signalų apdorojimo aparatūrą panaudojant PLIS, būtų susipažinę su pagrindiniais projektavimo metodais ir orientuotųsi šiuolaikinėje elementinėje bazėje, bei programiniuose paketuose.

3.1 PROGRAMUOJAMŲ LOGINIŲ INTEGRINIŲ SCHEMŲ KLASIFIKAVIMAS

Suklasifikuokime PLIS[8,9] pagal struktūrą, nes toks klasifikavimas gerai perteikia bendrą vaizdą apie užduoties klasę, tinkančią vienokiam ar kitokiam PLIS sprendimui. Bendrai priimta PLIS vertinti pagal jos loginę talpą, tai yra 2IR-NE ekvivalentinių ventilių skaičiumi reikalingu tam tikram projektui ant bazinio PLIS kristalo realizuoti. Žinoma šis vertinimas yra sutartinis, kadangi PLIS nesudaryta iš 2IR-NE ventilių tiesioginėje formoje, bet norint palyginti skirtingų architektūrų PLIS šis būdas tinka. Pagrindinis tokios klasifikacijos kriterijus yra tas, kad egzistuoja skirtingų tipų ir komutacijos būdų loginių elementų matricos. Pagal šį požymį galima išskirti keletą PLIS klasių.

Programuojamuos loginės matricos — tradiciškiausias PLIS tipas, apimantis IR ir ARBA programuojamas matricas. Šiai klasei priklauso FPLA (Field Programmable Logic Array) ir FPLS (Field Programmable Logic Sequencers). Tokių PLIS pavyzdžiais gali būti rusiškos mikroschemos К556РТ1, РТ2, РТ21. Tokios architektūros trūkumas yra ne pilnai išnaudojami ARBA programuojamos matricos resursai, todėl toliau buvo vystoma programuojamų loginių matricų architektūra PAL (Programmable Array Logic) — tai PLIS turinčios programuojamą IR matricą ir fiksuota ARBA matricą. Šiai klasei priskiriama dauguma šiuolaikinių PLIS su ne dideliu integracijos laipsniu. Šios klasės atstovai yra šios КМ1556ХП4, ХП6, ХП8, ХЛ8 ir 1980 metais

[8] Шипулин С.Н., Храпов В.Ю. Особенности проектирования цифровых схем на ПЛИС // Chip News. — 1996. — № 5. — С. 40–43.
[9] Шипулин С.Н., Храпов В.Ю. Основные тенденции развития ПЛИС // Электронные компоненты. — 1996. — № 3-4. — С. 26.

INTEL, ALTERA, AMD, LATTICE ir kitų firmų gamintos PLIS mikroschemos. Kitas tradicinis PLIS tipas — programuojama makrologika. Jos išskirtinė savybė ta, kad ji turi tik vieną programuojamą matricą IR-NE arba ARBA-NE, bet dėl didelio inversinių grįžtamųjų ryšių skaičiaus gali suformuoti sudėtingas logines funkcijas. Šiai klasei priklauso PLHS501 ir PLHS502 firmos SIGNETICS mikroschemos turinčios IR-NE matrica, o taip pat XL78C800 firmos EXEL, pagaminta ARBA-NE matricos pagrindu.

Paminėtosios PLIS architektūros turi mažą ląstelių skaičių, ir šiuo metu jau yra morališkai pasenusios ir naudojamos tik labai paprastuose įrenginiuose, kuriems nėra gaminamų vidutinio integracijos laipsnio integrinių mikroschemų. Žinoma, kad skaitmeniniam signalų apdorojimui jos netinkamos.

3.2 CPLD IR FPGA ARCHITEKTŪROS

PLD (Programmable Logic Device) mikroschemų architektūra yra pritaikyta skaitmeninių automatų realizavimui. Šios pažangios architektūros privalumas yra tai, kad programuojami komutuojami matricų blokai — tai PLIS, sudaryta iš kelių matricinių loginių blokų apjungtų komutuojamąja matrica. Kiekvienas matricinis loginis blokas sudaro PLD struktūrą, tai yra programuojamą matricą IR, fiksuota matrica ARBA ir makroląstelės. PLIS programuojamų komutuojamų matricų blokų tipo, kaip taisyklė turi didelį integracijos laipsnį (iki 10000 ekvivalentinių ventilių, iki 256 makroląstelių). Šiai PLIS klasei priklauso MAX5000 ir MAX7000 šeimos mikroschemos ALTERA firmos, XC7000 ir XC9500 šeimos mikroschemos XILINX firmos, o taip pat daugelis kitų mikroschemų galinamų Atmel, Vantis, Lucent ir kitose firmose. Šios klasės mikroschemos vadinamos CPLD (Complex Programmable Logic Device).

Kitas PLIS architektūros tipas — programuojamos ventilinės matricos, sudarytos iš loginių blokų ir komutavimo kelių — programuojami matricų sujungimai. Tokių PLIS loginiai blokai sudaryti iš vieno arba kelių sąlyginai paprastų loginių elementų, kurių pagrindas yra perkodavimo lentelė (Look-up table — LUT), programuojamas multipleksorius, D-trigeris, o taip pat valdymo grandinė. Tokių paprastų elementų gali būti pakankamai daug, pavyzdžiui šiuolaikinės PLIS kurių loginė talpa siekia iki 1 mil. ventilių yra keliasdešimt tūkstančių tokių elementų. Dėl tokio didelio loginių elementų skaičiaus jie turi ženklų trigerių skaičių, o kai kurios PLIS šeimos turi integruotus perkonfigūruojamus atminties modulius (embedded array block — EAB), kas šios architektūros PLIS padaro labai patogiu įrankiu skaitmeninių signalų apdorojimo algoritmų realizavimui. Pagrindinės šių algoritmų operacijos: perdauginimas, daugyba iš konstantos, sumavimas ir signalo užlaikymas. Viskas būtų gerai, bet kombinacinė šių PLIS dalis yra apribota, todėl kartu su programuojamom ventilinėm matricom naudojamas CPLD valdymo ir interfeiso realizavimui.

Šiomis savybėmis pasižyminčios PLIS vadinamos FPGA (Field Programmable Gate Array). FPGA klasei priklauso XC2000, XC3000, XC4000, Spartan, Virtex firmos XILINX; ACT1, ACT2 firmos ACTEL, o taip pat FLEX8000 šeimą gaminama firmos ALTERA, ir dar kai kurios firmų Atmel ir Vantis PLIS.

Tipinis FPGA PLIS pavyzdys yra Spartan mikroschemų šeima gaminama firmos XILINX.

Daug konfigūruojamų loginių blokų (Configurable Logic Blocks — CLBs) yra apjungiami apjungiamąja matrica. FPGA architektūrai būdingi įvedimo/išvedimo elementai (input/output blocks — IOBs), kurie leidžia realizuoti dvikryptį įvedimą išvedimą, trečią būsena ir t.t.

Šiuolaikinės PLIS turi mazgų testavimo galimybę per JTAG (B-scan), o taip pat turi vidinį generatorių (Osc) ir nuoseklaus konfigūravimo valdymo schemą

Firma Altera tobulindama FPGA architektūrą pasiūlė FLEX10K mikroschemų šeimoje taip vadinamą dviejų lygių apjungiamosios matricos architektūrą.

Loginiai elementai apjungiami į grupes vadinamas loginiais blokais. Loginio bloko viduje loginiai elementai sujungiami su lokalia programuojama jungiamąja matrica, kuri leidžia sujungti bet kurį loginį elementą su bet kuriuo. Loginiai blokai sujungti tarpusavyje ir su įvedimo išvedimo/elementais per globalią programuojamąją sujungiančiąja matrica. Lokalioji ir globalioji jungiamosios matricos turi nepertraukiamą struktūrą — kiekvienam sujungimui išskiriamas nepertraukiamas kanalas.

Toliau tobulinant dabartinę architektūra yra siekiama sukurti kombinuotą architektūra, kuri būtų dar patogesnė skaitmeninių signalų apdorojimo algoritmams realizuoti panaudojant perkodavimo lenteles ir perkonfigūruojamus atminties modulius, charakteringas FPGA struktūras ir daugialyges PLIS su patogiu skaitmeninių automatų realizavimu CPLD architektūroje. APEX20K PLIS Altera firmos gaminyje yra loginiai elementai visų išvardintų tipų, o tai leidžia panaudoti PLIS kaip elementarųjį elementą SOC (system-on-chip) kūrimui. Pagrindinė SOC idėja: visos elektroninės schemos integravimas viename kristale (pavyzdžiui personaliniame kompiuteryje toks SOC galėtu apjungti procesorių, atmintine, ir t.t.). Tokių sistemų komponentai kuriami atskirai ir saugomi parametrizuotuose moduliuose falų pavidale. Galutinė SOC mikroschemos struktūra realizuojama pasitelkus virtualių komponentų baze panaudojant projektavimo automatizavimo sistemos programą, elektroniniams įrenginiams — EDA (Electronic Design Automation). Dėl naudojamos vieningos virtualių komponentų standartizacijos sistemos, galima naudoti skirtingų gamintojų virtualius komponentus toje pačioje sistemoje.

3.3 PROGRAMUOJAMOS LOGINĖS INTEGRINĖSE SCHEMOS PASIRINKIMAS

Renkantis elementinę baze projektuojamai signalų apdorojimo sistemai vadovaujamasi šiais kriterijais:

1. Greitaveika;
2. Loginė talpa, kurios užtektų sukurtam algoritmui realizuoti;
3. Schemotechniniai ir konstrukciniai PLIS parametrai, patikimumas, darbinės temperatūros diapazonas, atsparumas jonizuojančiam spinduliavimui ir t.t.;
4. Projekto realizavimo priemonių kaina, įskaitant programinio aprūpinimo, bei aparatūrinių derinimo priemonių kaina;
5. PLIS programatoriaus arba konfiguracinės ROM kaina;
6. Metodinio ir techninio palaikymo egzistavimas;
7. Tiekėjų egzistavimas, bei jų patikimumas šalyje;
8. Mikroschemos kaina;

Remiantis šiais kriterijais panagrinėkime lyderiaujančius PLIS gamintojus.

3.3.1 ALTERA FIRMOS GAMINAMŲ PLIS APŽVALGA

Firma Altera Corporation, (101 Inovation Drive, San Jose, CA 95134, USA, http://www.altera.com/) buvo įkurta 1983 metų birželio mėnesį. Šiuo metu naujausias šios firmos gaminys yra APEX20K mikroschemų šeima, kurios architektūrinės ypatybės jau aptartos, o lentelėje 3-1 pateikti pagrindiniai šios PLIS mikroschemų šeimos parametrai.

	EP20K100	EP20K160	EP20K200	EP20K300	EP20K400	EP20K600	EP20K1000
Maksimalus ekvivalentinių ventilių skaičius	263 000	404 000	526 000	728 000	1 052 000	1 537 000	2 670 000
Loginių elementų skaičius	4 160	6 400	8 320	11 520	16 640	24 320	42 240
Integruoti atminties blokai	26	40	52	72	104	152	264
Maksimali atminties apimtis bitais	53 248	81 920	106 496	147 456	212 992	311 296	540 672
Makroląstelių skaičius	416	640	832	1 152	1 664	2 432	4 224
Vartotojo išvadų skaičius	252	320	382	420	502	620	780

lentelė 3-1 paridinės APEX20K PLIS šeimos charakteristikos (firmos ALTERA)

Be šios šeimos mikroschemų ALTERA gamina CPLD šeimas MAX3000, MAX7000, MAX9000 (pasenusios serijos specialiai neminimos) ir FPGA šeimas FLEX10K, FLEX8000, FLEX6000. Teigiamas faktorius renkantis Alteros PLIS yra tai, kad automatizavimo ir projektavimo sistemos programinį paketą MAX+PLUS II BASELINE ver. 9.3 iš http://www.altera.com/ tinklapio galima parsisiųsti nemokamai ar gauti CD Altera Digital Library kuriame yra visas dokumentacijos rinkinys apie PLIS architektūra ir panaudojimą. lentelė 3-2 lentelėje pateiktos pagrindinės MAX+PLUS II BASELINE ver. 9.3 paketo savybės.

Palaikomi įrenginiai	EPF10K10, EPF10K10A, EPF10K20, EPF10K30, EPF10K30A, EPF10K30E (iki 30 000 ekvivalentinių ventilių), EPM9320, EPM9320A, EPF8452A, EPF8282A, MAX7000, FLEX6000, MAX5000, MAX3000A, Classic
Projekto aprašymo priemonės	Schematinis įvedimas, AHDL palaikymas, trečiųjų firmų projektavimo ir automatizavimo paketų interfeiso palaikymas, topologinis redaktorius, hierarchinė projekto struktūra, parametrizuotų modulių biblioteka.
Projekto kompiliavimo priemonės	Loginė sintezė ir trasavimas, automatinis klaidų aptikimas, mega funkcijų palaikymas programose MegaCore ir AMPP.
Projekto testavimo priemonės	Laikinė analizė, funkcinis ir laikinis modeliavimas, signalų analizė, galimybė panaudoti trečių firmų modeliavimo (simuliavimo) programas.

lentelė 3-2 pagrindinės MAX+PLUS II BASELINE ver. 9.3 paketo savybės

Firmos Altera PLIS pasižymi programavimo sistemoje galimybe. Konfiguracinių duomenų į mikroschemą įrašymui yra pateikiamos duomenų persiuntimo kabelių schemos ByteBlaster ir ByteBlasteMV. Altera PLIS gamina komerciniam ir industriniam temperatūrų diapazonams.

3.3.2 XILINX KOMPANIJOS PLIS GAMINIŲ APŽVALGA

Kompanija Xilinx Inc. (2100 Logic Drive, San Jose, CA 95124-3400, USA, http://www.xilinx.com/ buvo įsteigta 1984 metų vasario mėnesį, jos naujausias PLIS gaminys yra Virtex šeima, kurios pagrindinės charakteristikos patektos lentelėje 3-3.

	XCV50	XCV100	XCV150	XCV200	XCV300	XCV400	XCV600	XCV800	XCV1000
Maksimalus ekvivalentinių ventilių skaičius	57 906	108 904	164 674	236 666	322 970	468 252	661 111	888 439	1 124 022
Loginių elementų skaičius	1 728	2 700	3 888	5 292	6 912	10 800	15 552	21 168	27 648
Maksimali atminties apimtis bitais	24 576	38 400	55 296	75 264	98 304	153 600	221 184	301 056	393 216
Vartotojo išvadų skaičius	180	180	260	284	316	404	512	512	512

lentelė 3-3

Šeimos Virtex PLIS pasižymi, plataus spektro greitaeigiais trasavimo resursais, atskirai išskirtu blokiniu RAM, patobulinta pagreitinto pernešimo logika. Šios serijos PLIS šeimoje pasiektas didelis mainų greitis kristale — iki 200Mhz (standartas HSTL IV). Virtex serijos kristalai dėl išvystytos gamybos technologijos ir patobulinto tikrinimo proceso, palyginus nedaug kainuoja (iki 40% nuo ekvivalentinės XC4000XL serijos kainos).

Be Virtex šeimos, Xilinx gamina FPGA šeimas XC3000A, XC4000E, Spartan, XC5200, o taip pat CPLD XC9500 ir mažai energijos vartojančią serija CoolPLD.

Egzistuoja nemokamas automatizavimo ir projektavimo sistemos programinis paketas — WebPACK kuris palaiko CPLD XC9500, CoolPLD ir Virtex, įvedimo aprašymo algoritmas rašomas VHDL kalba.

Reikėtų paminėti, kad Xilinx atnaujino eilę CPLD gaminių bei programinę įrangą, kuri šiuo metu tobulinama kartu su Synopsys firmos pagalba.

PLIS Xilinx gamina komerciniame ir industriniame temperatūrų diapazone su karinio ir kosminio panaudojimo galimybėmis.

3.3.3 ACTEL KORPORACIJOS PLIS GAMINAMŲ APŽVALGA

Actel korporacija (955 East Arques Avenue, Sunnyvale, CA 94086-4533, USA, http://www.actel.com/) buvo įsteigta 1985 metais. Actel gaminamų PLIS ypatybė yra naudojamoje technologijoje Antifuse-technology, kuri programavimo metu sudaro metalizuotą trumpiklį. Ši technologija suteikia didelį patikima ir lankstų trasavimą, be to ne reikalinga konfiguracinė ROM. Pagal šią technologija gaminamos šeimos ACT1, ACT2, 1200XL, o taip pat naujos šeimos 54SX, A40MX, ir A42MX (su integruotais atminties moduliais), pasižymi geru kainos ir loginės talpos santykiu.

Naujoji firmos Actel šeima ProASIC turi iki 500000 ekvivalentinių loginių ventilų. Dėl panaudotos FLASH technologijos ir į kristalą integruotos atmintinės ši šeima yra energetiškai nepriklausoma.

Projektuojamiems Actel firmos PLIS iki 2000-01-31 buvo nemokamai platinamas programinis paketas Actel DeskTOP, turintis projekto įvedimo priemones, modeliavimą, projekto testų generatorių VeriBest ir Synplicity sintezės priemones. Turbūt Actel DeskTOP programinis paketas yra geriausias iš nemokamų.

Deja Actel mikroschemoms gaminamoms pagal Antifuse-technologiją reikalingas specialus programatorius, kuris pakankamai brangiai kainuoja. Iš kitos pusės šių mikroschemų didelis patikimumas yra labai perspektyvi savybė specializuotiems projektams. RH1280 serijos PLIS turi šias charakteristikas:

1. Leistina apspinduliavimo dozė 300000 rad;
2. Loginė talpa 16000 ekvivalentinių ventilių.
3. Greitaveika iki 135 Mhz.

Šio tipo PLIS buvo panaudotos skaitmeniniams vaizdams Marse apdoroti, valdymo sistemose, bei signalo siuntimo į Žemę įrenginyje. Dabar gaminamos naujos atsparios radiacijai PLIS šeimos.

PLIS Actel gamina komerciniame ir industriniame temperatūrų diapazone su karinio ir kosminio panaudojimo galimybėmis.

4. PERPROGRAMUOJAMŲJŲ SISTEMOJE CPLD ŠEIMOS XC9500 MIKROSCHEMOS

4.1 ŠEIMOS YPATYBĖS

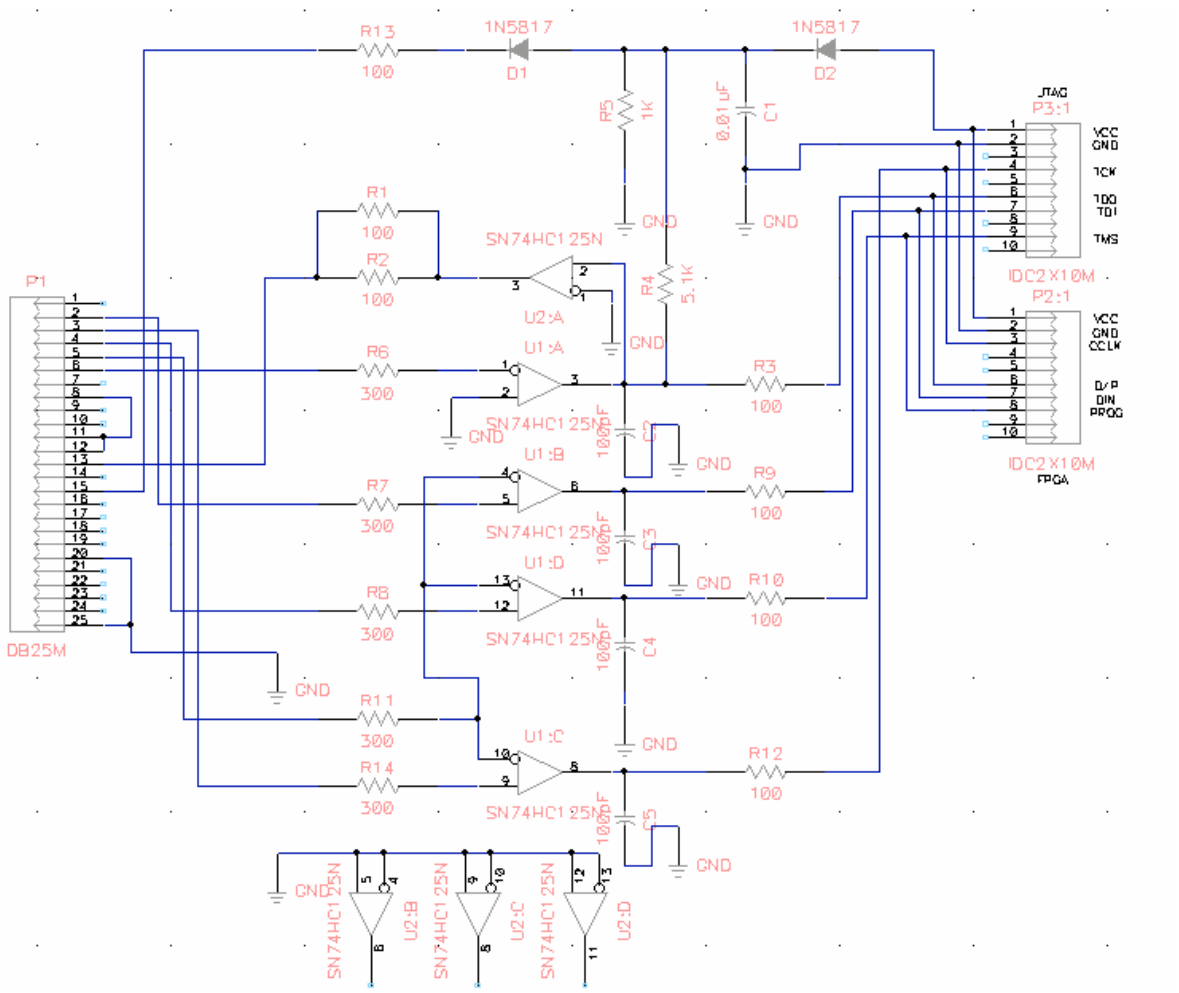
- Didelė greitaveika.
    - Vėlinimas nuo įėjimo iki išėjimo, visiems išvadams, neviršija 5 ns.
    - 16-iolikos skilčių skaitliuko darbo dažnis iki 125Mhz.
- Platus šios šeimos mikroschemų pasirinkimas pagal integracijos laipsnį.
    - Nuo 36 iki 288 makroląstelių, arba nuo 800 iki 6400 ventilių.
- Perprogramavimo galimybė sistemoje prie 5V maitinimo įtampos.
    - Nemažiau kaip 10000 įrašymo/ištrynimo ciklų.
    - Programavimas/trynimas visame komerciniame maitinimo įtampų ir temperatūrų diapazone.
- Išvadų blokavimo galimybė prieš trasavimą.
- Lankstus 36V18 funkcinis blokas.
    - Bet kuri iš 18 funkcinio bloko makroląstelių gali vykdyti 36 kintamųjų (nuo 1 iki 90 termų) loginę funkciją.
    - Globalūs ir programuojamieji taktiniai signalai, išėjimo leidimo signalai, trigerio nustatymo ir perkrovimo signalai.
- Programuojamas sumažintos vartojamos galios rėžimas kiekvienoje makroląstelėje atskirai.
- Signalo užlaikymo valdymas bet kuriame iš išėjimų.
- Galimybė užprogramuoti bendrojo išvado paskirtį.
- Schemos apsaugos nuo kopijavimo galimybė.
- Galingas išėjimas (24 mA), galintis veikti prie 3,3V arba 5V išėjimo kaskadų maitinimo įtampos.
- Pilnas periferijos skanavimo palaikymas pagal IEEE Std 1149.1 (JTAG) standartą.
- Gaminamos pagal CMOS 5V FastFLASH technologija.
- Galimybė lygiagrečiai programuoti keletą XC9500 šeimos mikroschemų.

4.2 XC9500 ŠEIMOS MIKROSCHEMŲ APŽVALGA

XC9500 šeima turi CPLD (Complex Programmable Logic Device) – kompleksiškai programuojamo loginio įrenginio struktūra. CPLD struktūra sudaryta iš makroląstelių, kurios leidžia sudaryti kelių kintamųjų logines funkcijas, kintamųjų skaičių riboja tik termų skaičius. Šio tipo mikroschemas galima panaudoti nestandartiniams ALĮ (Aritmetinis Loginis Įrenginys), dešifratoriams, multipleksoriams ir kt. kurti, ten kur reikalingos daugelio kintamųjų funkcijos ir nedaug trigerių.

Ši mikroschemų šeima gali būti naudojama masinės gamybos aparatūroje, o taip pat sistemose kurioms būtina perprogramavimo sistemoje galimybė.

XC9500 šeimos mikroschemoms ne reikalingas specialus programatorius, nes programavimas vyksta prie 5V maitinimo įtampos per specialius mikroschemos išvadus (JTAG prievadą pav. 4-1) toje pačioje sistemoje kur ir naudojama ši CPLD. Minimalus mikroschemos perprogramavimų ciklų skaičius viršija 10000, o įrašyta konfigūracija gali būti saugoma daugiau kaip 20 metų.



pav. 4-1

XC9500 mikroschemų šeima susideda iš šešių mikroschemų, kurių makroląstelių skaičius nuo 36 iki 288 (nuo 800 iki 6400 ventilių, atitinkamai) skirtinguose korpusuose. Visose šios šeimos mikroschemose palaikomas kontaktinis išvadų suderinamumas, kuris suteikia lanksčia perėjimo galimybe nuo vienos prie kitos mikroschemos tokiame pat korpuse.

Lentelėje 4-1 pateikti pagrindiniai XC9500 mikroschemų šeimos parametrai, o lentelėje 4-2 visi gaminami korpusai su nurodytu naudotinų išvadų skaičiumi.

	XC9536	XC9572	XC95108	XC95144	XC95216	XC95288
Makroląstelių skaičius	36	72	108	144	216	288
Ventilių skaičius	800	1600	2400	3200	4800	6400
Trigerių skaičius	36	72	108	144	216	288
t_{PD}[ns]	5	7,5	7,5	7,5	10	10
t_{SU}[ns]	3,5	4,5	4,5	4,5	6,0	6,0
t_{CO}[ns]	4,0	4,5	4,5	4,5	6,0	6,0
f_{CNT}[MHz]	100	125	125	125	111,1	111,1
f_{SYSTEM}[MHz]	100	83,3	83,3	83,3	66,7	66,7

Lentelė 4-1 pagrindiniai XC9500 mikroschemų šeimos parametrai

Pastaba:

f_{CNT}[MHz] — 16-likos skilčių skaitliuko darbo dažnis

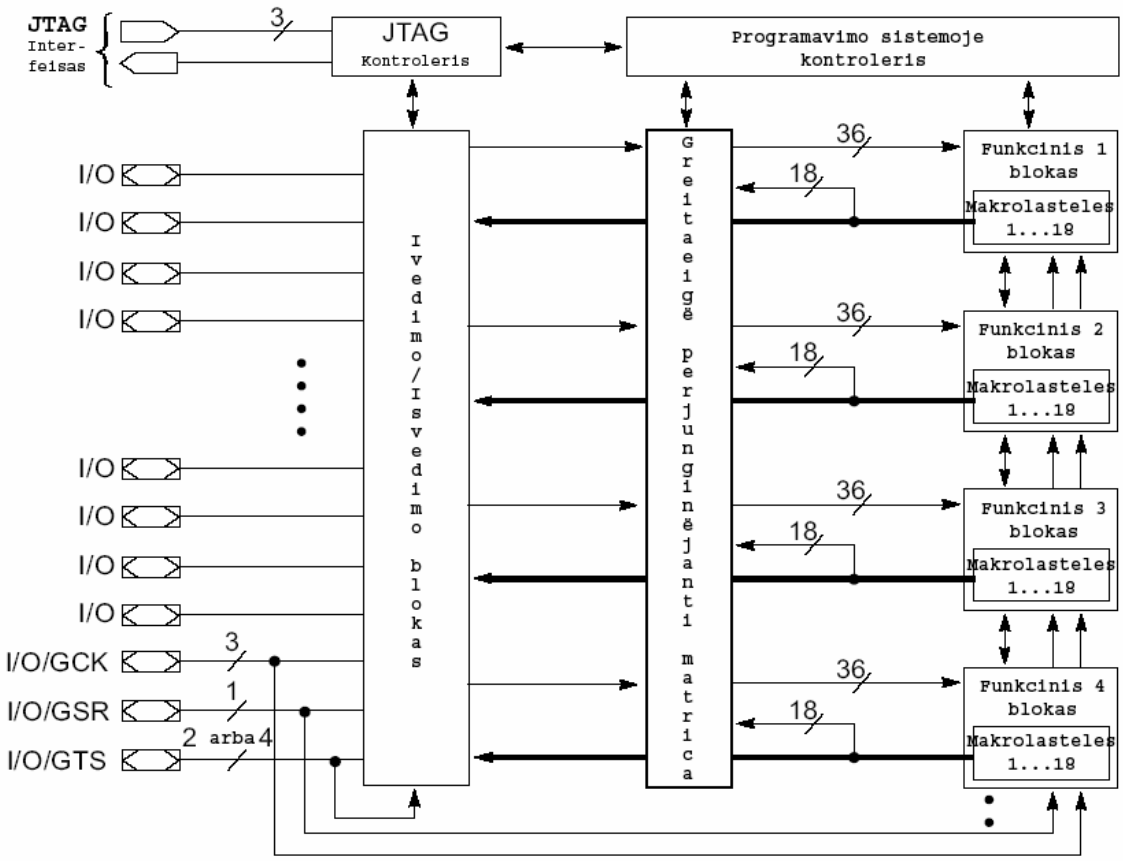
f_{SYSTEM}[MHz] — vidinis projektų dažnis, naudojat keletą funkcinių blokų.

	XC9536	XC9572	XC95108	XC95144	XC95216	XC95288
VQFP-44	34					
PLCC-44	34	34				
CSP-48	34					
PLCC-84		69	69			
TQFP-100		72	81	81		
PQFP-100		72	81	81		
PQFP-160			108	133	133	
HQFP-208					166	168
BGA-352					166	192

Lentelė 4-2 visi gaminami korpusai su nurodytu naudotinų išvadų skaičiumi

4.3 XC9500 ŠEIMOS MIKROSCHEMŲ ARCHITEKTŪRA

Kiekviena XC9500 šeimos mikroschema sudaryta iš posistemių, kurios susideda iš daugelio *funkcinių blokų* ir *įvedimo – išvedimo blokų*, sujungtų su *perjunginėjančia matrica* pav. 4-2.



Pav. 4-2

Įvedimo – išvedimo blokas buferizuoja visus mikroschemos įėjimus ir išėjimus. Kiekvienas *funkcinis blokas* sudarytas iš 18 makroląstelių su 36V1 struktūra, o tai leidžia sudaryti 18 loginių funkcijų, praktiškai pagal bet kokia 36 kintamųjų kombinaciją. *Perjunginėjančioji matrica* valdo bet kokį išėjimo ir įėjimo signalų siuntimą tiek iš ir tiek į *funkcinį bloką*. Nuo 12 iki 18, kiekvieno *funkcinio bloko* išėjimo signalų (priklausomai nuo išvadų kiekio korpuse) ir atitinkami išėjimo leidimo signalai patenka tiesiogiai į *įvedimo – išvedimo bloką.*

Visus CPLD šeimos XC9500 išvadus galima suskirstyti į tris grupes:

1. JTAG prievado išvadai, per kuriuos vykdomas periferinis skanavimas ir programavimas.
2. Loginiai išvadai kurie gali atlikti įėjimų, išėjimų arba mišrią įėjimo – išėjimo (I/O) funkcija.
3. Valdantieji išvadai į kuriuos paduodami globalus valdymo signalai tokie kaip sinchronizacijos signalas (GCK), nustatymo/perkrovimo (GSR) ir trečios būsenos valdymo signalas (GTS). Valdantieji išvadai gali atlikti ir loginių išvadų funkcijas.

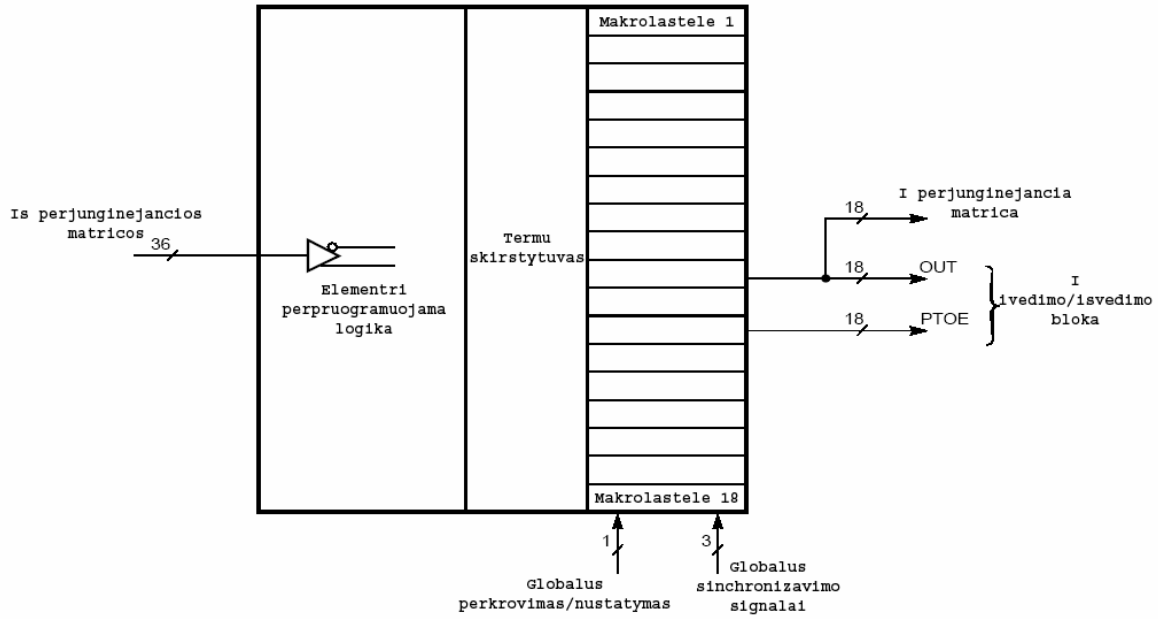
Taip pat yra bendras ir maitinimo išvadai, bet atskirai yra maitinami *įvedimo – išvedimo blokai* (nuo maitinimo įtampos V_{CCIO}) o visi likusieji (nuo maitinimo įtampos V_{CCINT}).

4.3.1 FUNKCINIS BLOKAS

Funkcinis blokas pav. 4-3 sudarytas iš 18-likos nepriklausomų makroląstelių, kiekviena iš jų valdo kombinacinę ir/arba registrinę funkcija. Be to į *funkcinį bloką* ateina išėjimo leidimo, nustatymo/perkrovimo ir globalus sinchronizacijos signalai. Kiekvienas funkcinis blokas formuoja 18 išėjimo signalų, kurie patenka į *perjunginėjančią matricą* bei į *įvedimo – išvedimo bloką.*

Funkcinio bloko viduje yra loginė matrica sudaryta iš pirminių loginių vienetų (termų). Trisdešimt šeši įėjimai leidžia naudoti 72 tiesioginius ir invertuotus signalus, todėl šioje pirminėje loginėje matricoje galima suformuoti iki 90 termų. Bet koks šių termų poaibis gali būti pasiekiamas kiekvienai makroląstelei per termų paskirstymo schemą.

Visi *funkciniai blokai* turi vidinę grįžtamo ryšio grandį, kuri leidžia bet kokiam kiekiui, *funkcinio bloko*, išėjimo signalų patekti į savo nuosavą programuojamų elementų matricą, neišeinat už *funkcinio bloko* ribų.



Pav. 4-3

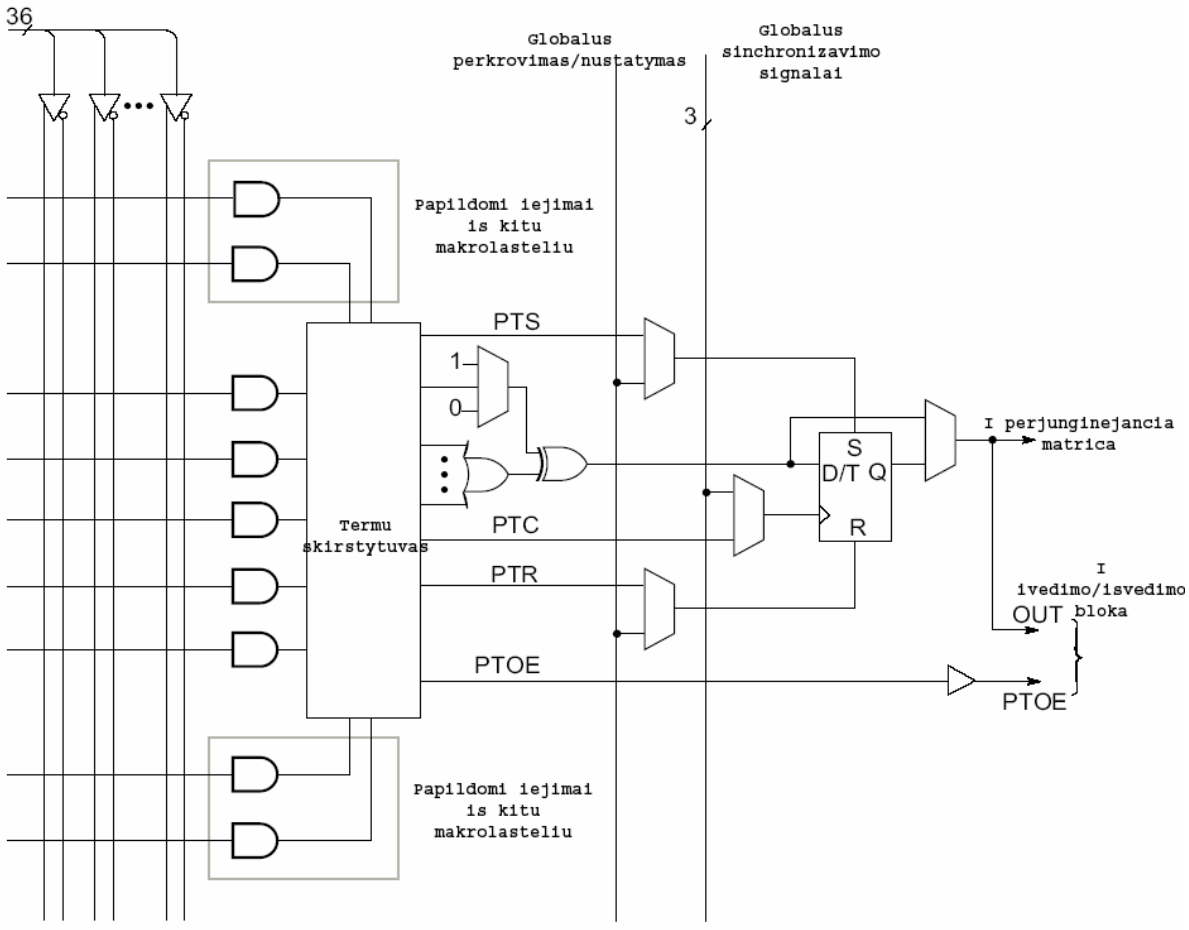
4.3.2 MAKROLĄSTELĖ

Bet kuri XC9500 mikroschemų šeimos makroląstelė gali atlikti loginę, kombinatorinę ir registrinę funkcijas. Makroląstelės struktūra kartu su programuojamąją, elementariųjų loginių (elementai IR, ARBA „termai“) operacijų, matrica priklausančia visam *funkciniam blokui*,

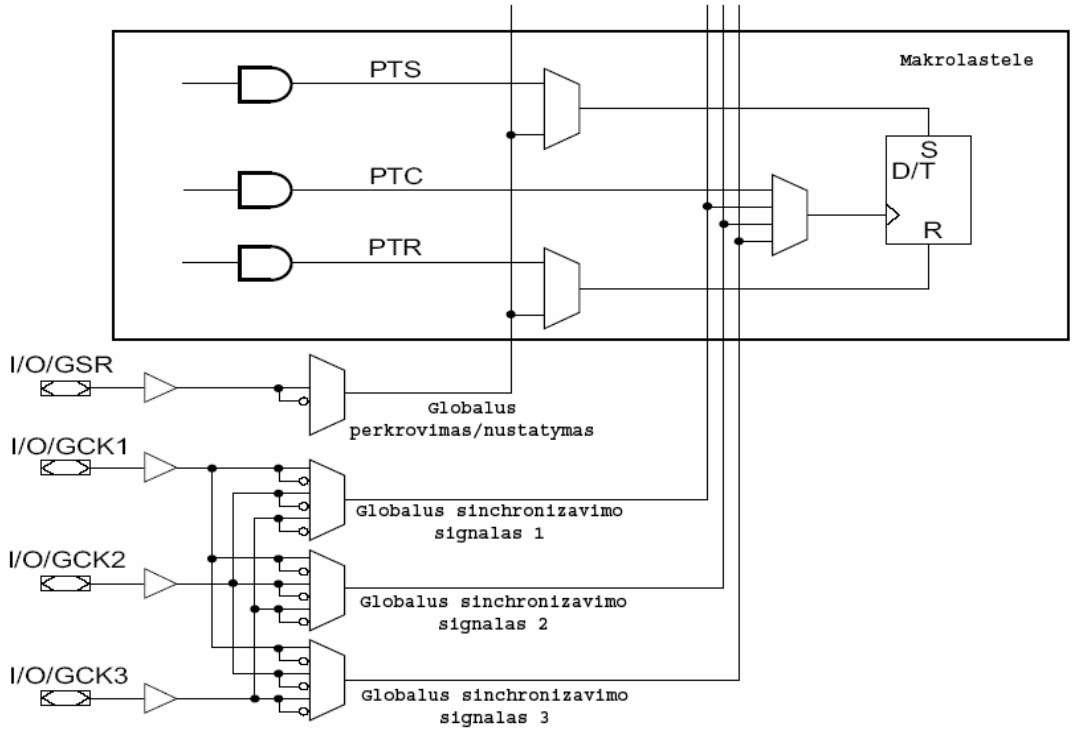
pavaizduota 3 pav. Kiekviena makroląstelė turi penkis pagrindinius ir keturis papildomus išėjimus patenkančius į termų skirstytuvą.

Iš elementariųjų loginių operacijų matricos penkios termos patenka į pagrindinius makroląstelės išėjimus ir gali būti panaudotos kombinatorinėms (ARBA ir griežtas ARBA) funkcijoms, arba kaip valdymo signalai, įskaitant ir įsimenančio elemento sinchronizacijos signalą PTC (Product Term Clock), taip pat jo nustatymo bei perkrovimo – PTS (Product Term Set) ir PTR (Product Term Reset), ir išėjimo leidimo PTOE (Product Term Output Enable) signalus. Per keturis papildomus įėjimus patenka signalai iš kitų makroląstelių. Įėjimo termų paskirtį tai arba kitai funkcijai atlikti, nustato kiekvienoje makroląstelėje esantis termų skirstytuvas.

Įsimenantis elementas makroląstelėje gali būti sukonfigūruotas kaip D-trigeris arba kaip T-trigeris arba gali būti visai nenaudojamas. Paskutiniu atveju loginės funkcijos signalas, jei reikia praleidžiamas tiesiogiai į kitas makroląsteles. Į kiekvieną trigerį gali būti paduodamas asinchroninio perkrovimo ir nustatymo signalas iš termų skirstytuvo. Mikroschemos įjungimo momentu visi registrai nustatomi į pradinę būseną, kuri užsiduodama vartotojui programuojant mikroschemą. Jeigu pradinė būsena neužprogramuota, tai registrai nustatomi į loginio „0“ lygį.



Pav. 4-4



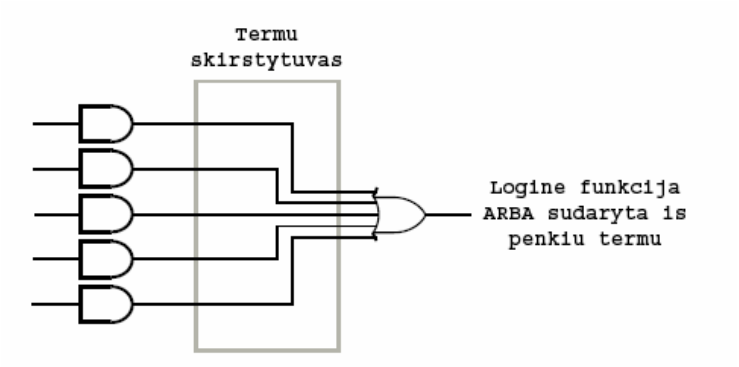
Pav. 4-5

Be to, kiekviena makroląstelė turi dar keturis globalaus valdymo signalus (trys sinchronizacijos – GCK1, GCK2, GCK3 – ir vienas perkrovimo/nustatymo – GSR signalas), kurie gali būti panaudojami trigerio darbo valdymui, kaip parodyta pav.4-5 . Globalūs valdymo signalai paimami tiesiogiai nuo mikroschemos valdymo išvadų, kurie šiaip jau gali būti panaudojami kaip loginiai įėjimai/išėjimai (I/O), nes šie išvadai taip pat sujungti su programuojamais įvedimo/išvedimo blokais.

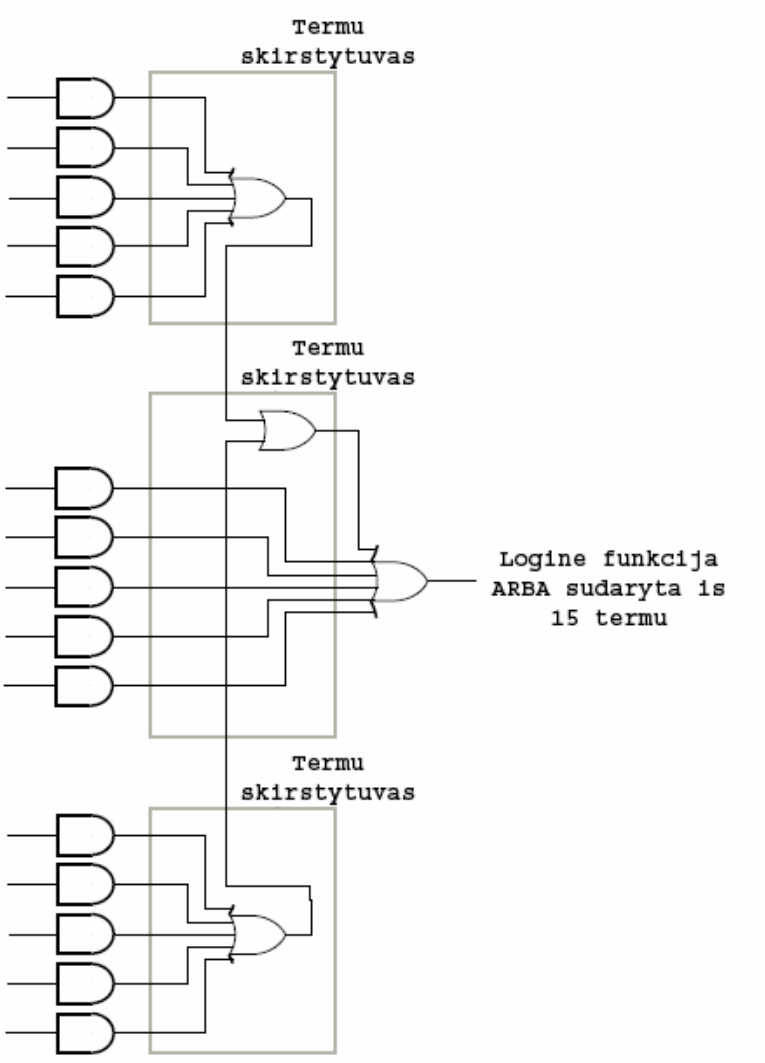
4.3.3 TERMŲ SKIRSTYTUVAS

Termų skirstytuvas tiesiogiai valdo penkių pirmųjų termų paskirtį kiekvienoje makroląstelėje. Pavyzdžiui, visos penkios termos gali būti priskiriamos ARBA elementui, kaip parodyta pav. 4-6.

Termų skirstytuvas gali pakeisti bet kurios termos paskirtį, *funkcinio bloko* viduje loginės talpos makroląstelėje praplėtimui daugiau kaip penkioms tiesioginėms termoms. Kiekvienai makroląstelei kuriai reikia papildomų termų, prieinama bet kuri iki tol nepanaudota terma kitoje makroląstelėje su sąlyga, kad abi makroląstelės priklauso tam pačiam *funkciniam blokui*. Vienai makroląstelei gali būti prieinamos iki 15 termų, kaip pavaizduota pav. 4-7.

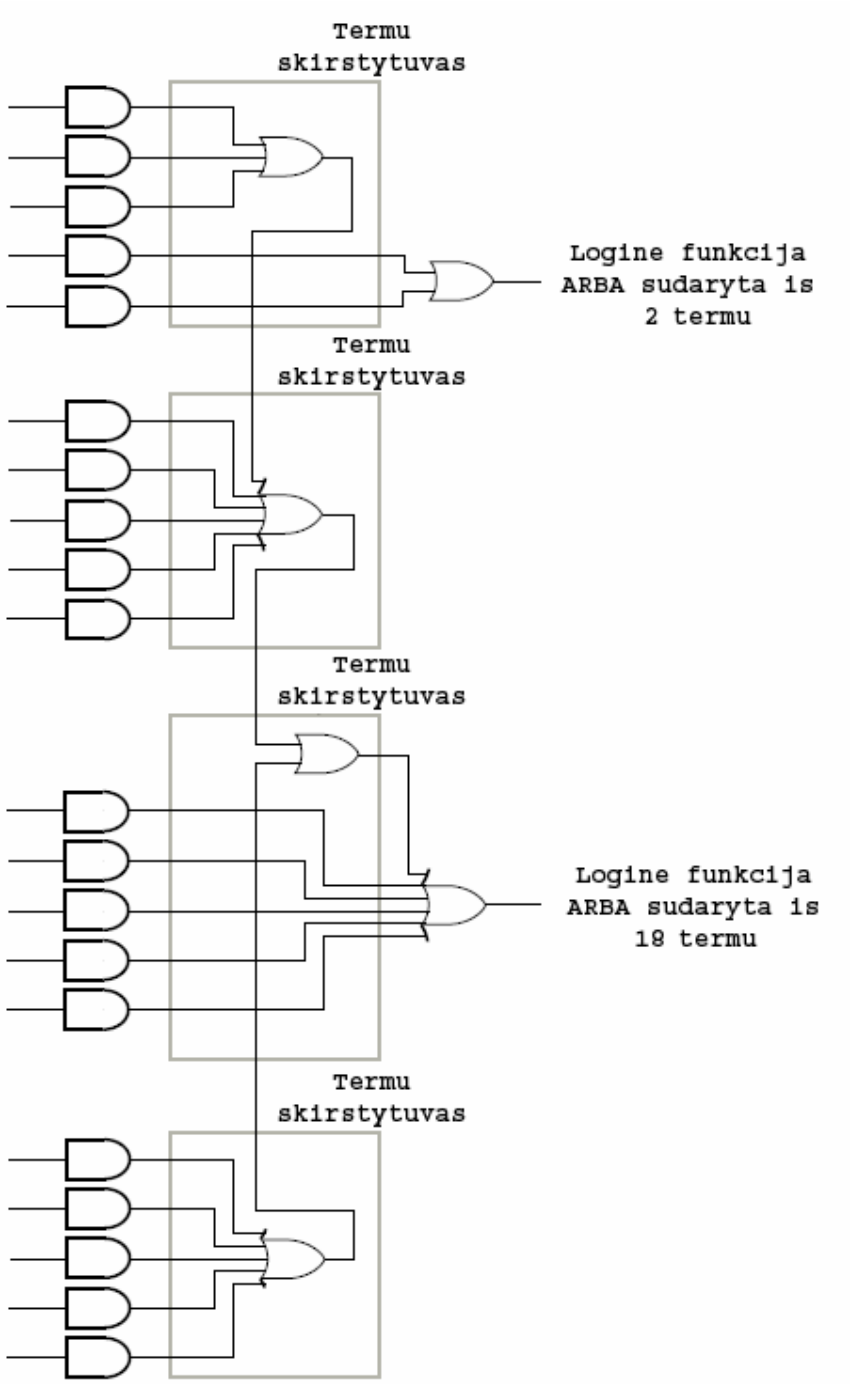


Pav. 4-6

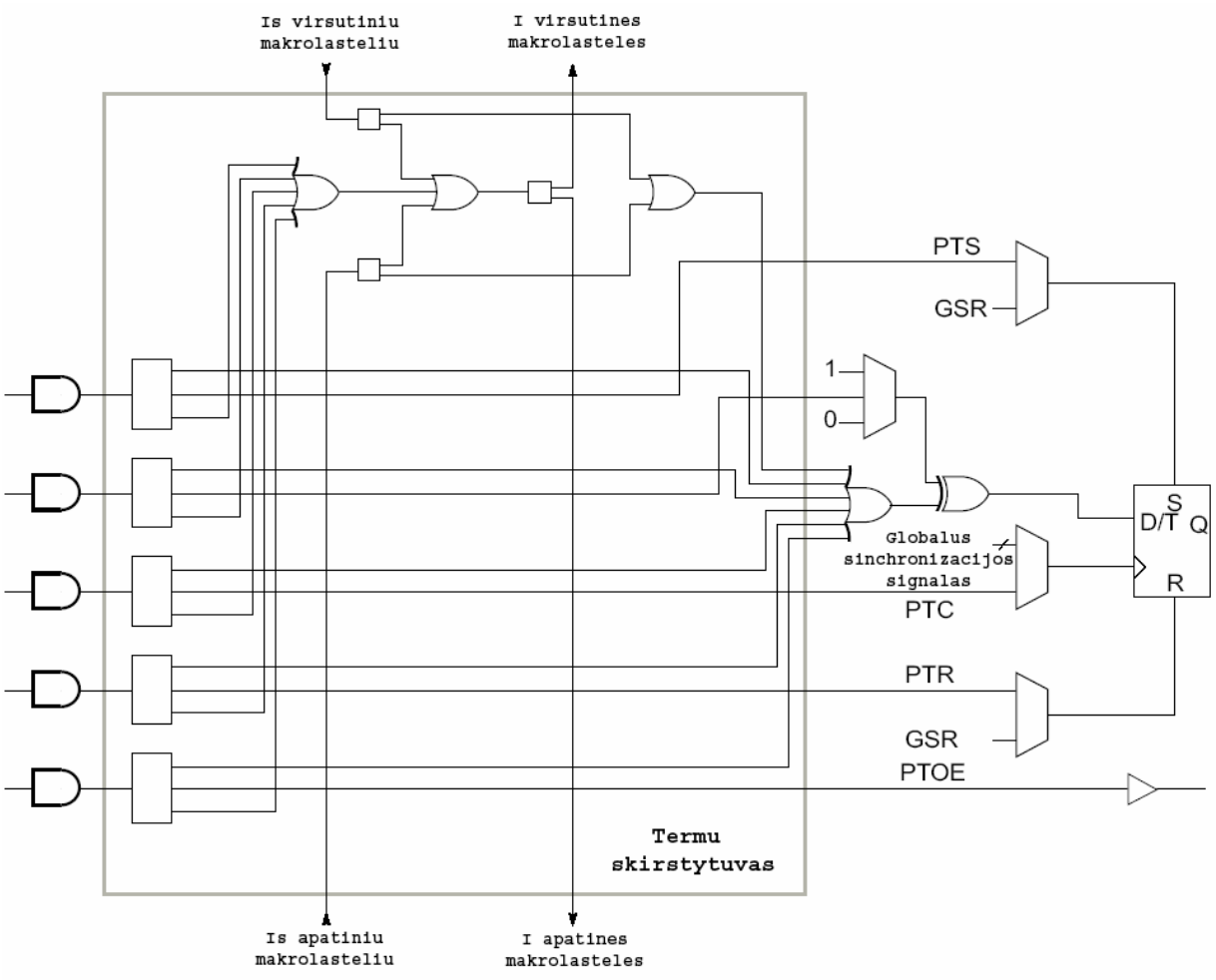


Pav. 4-7

Termų paskirstymo schema gali pakeisti bet kurios termos paskirtį iš bet kurios makroląstelės funkcinio bloko viduje, tai atliekama apjungimo būdu, apjungiant keletą makroląstelių kaip parodyta pav. 4-8


Termu
skirstytuvas
Logine funkcija
ARBA sudaryta is
2 termu
Termu
skirstytuvas
Termu
skirstytuvas
Logine funkcija
ARBA sudaryta is
18 termu
Termu
skirstytuvas

Pav. 4-8

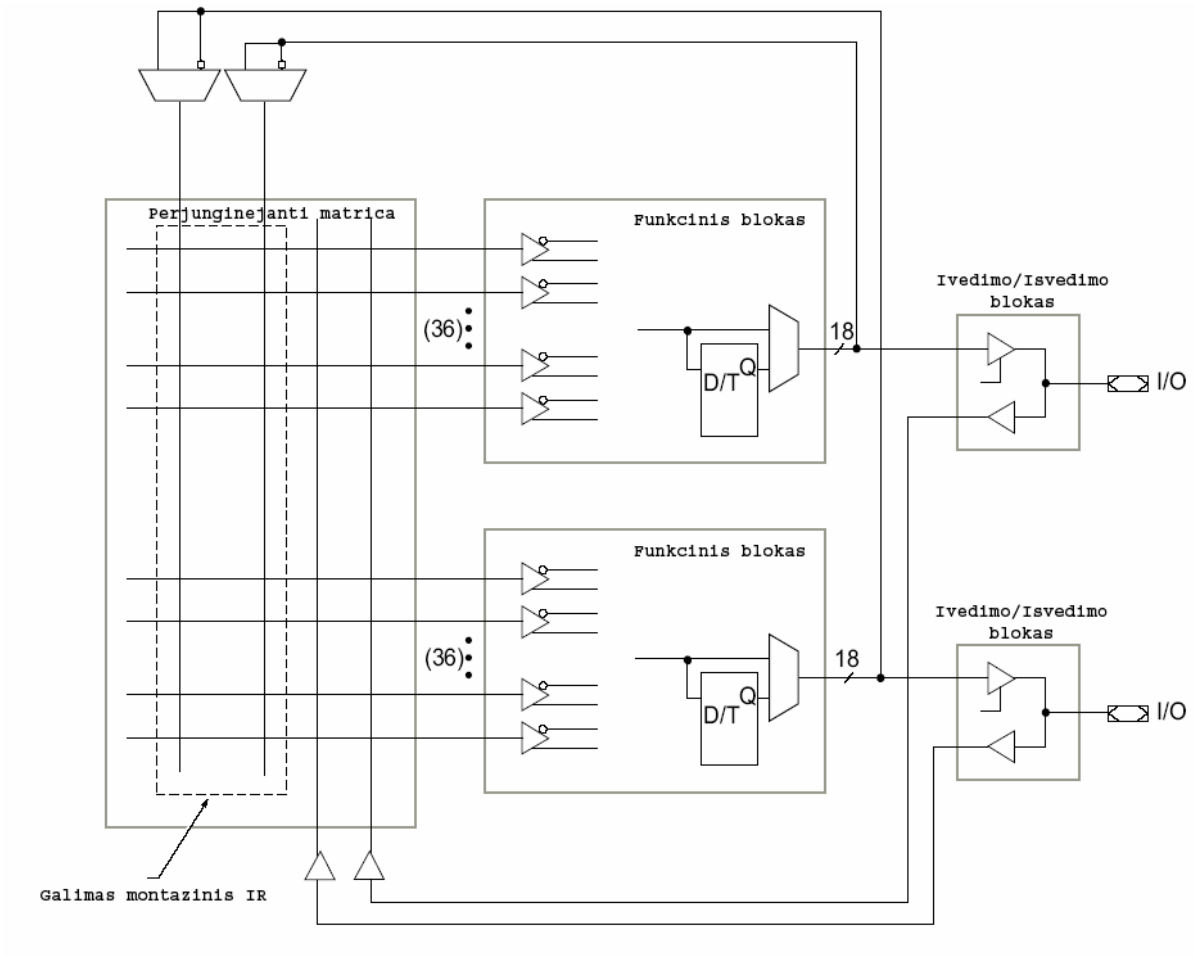


Pav. 4-9

Termų paskirstymo schemos darbo logika pavaizduota pav. 4-9. Čia stačiakampiai su keliais išėjimais šiame paveikslėlyje reiškia programuojamuosius komutatorius (selektorius) vieno įėjimo signalo į bet kurį iš išėjimų.

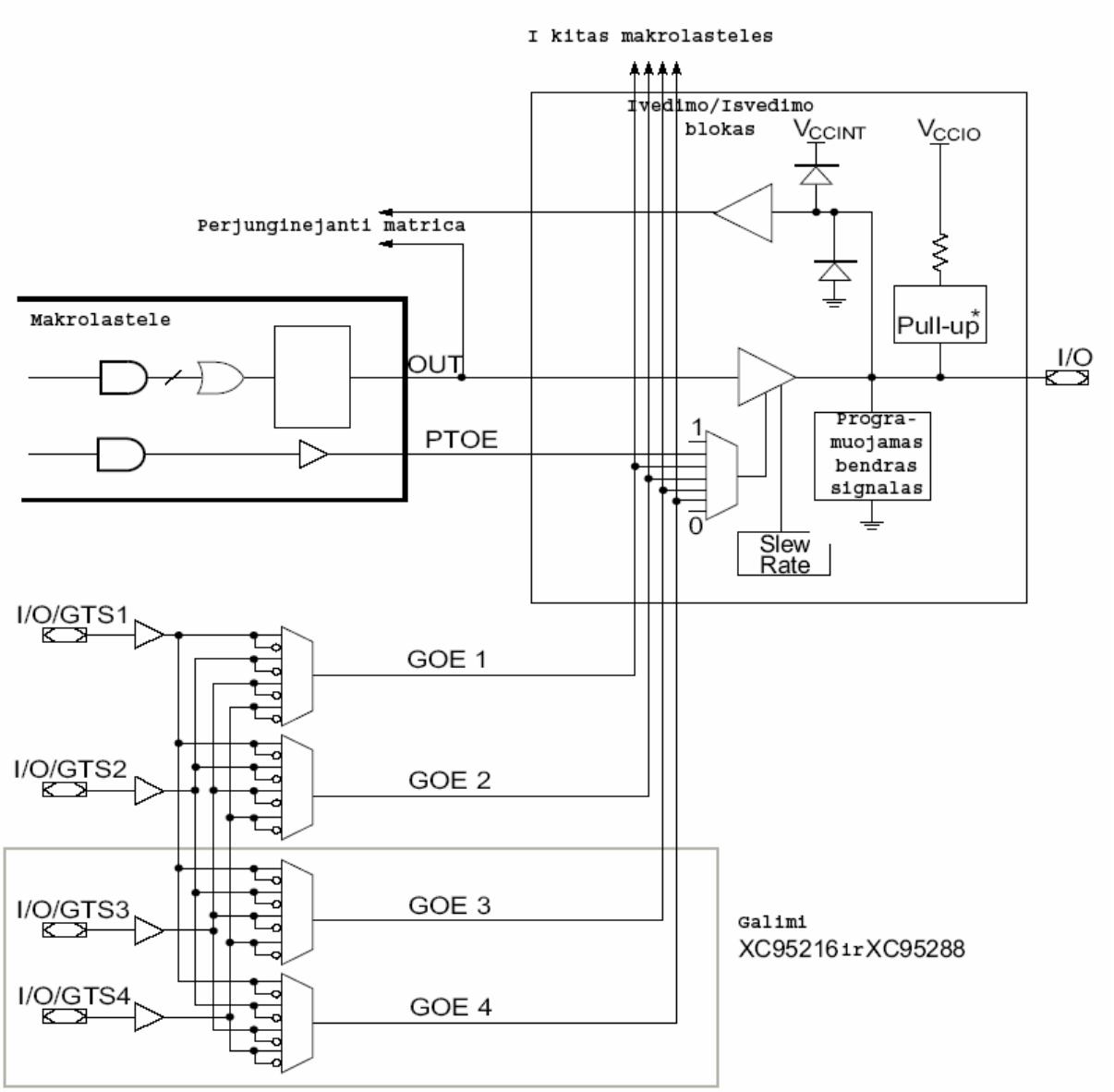
4.3.4 GREITAEIGĖ PERJUNGINĖJANTI MATRICA

Perjunginėjanti matrica vykdo signalų patenkančių iš *funkcinio bloko* išėjimų ir iš *įvedimo – išvedimo bloko* į funkcinio bloko išėjimus komutaciją, mikroschemos viduje, kaip parodyta pav.4-10. Esant būtinybei, *funkcinio bloko* išėjimo signalai gali būti apjungiami perjunginėjančios matricos viduje pagal schemą „montažinis IR“, tai praplečia logines galimybes ir leidžia padidinti darbinę apkrovą nutolusiems *funkciniams blokams*, be papildomų signalo vėlinimų. Ši savybė prieinama vidiniams sujungimams, imantiems signalą tik iš funkcinio bloko išėjimų, tai automatiškai panaudojama programiniame projektavimo pakete, kai tik įmanoma.


Perjunginejanti matrica
Funkcinis blokas
(36)
D/T
Q
18
Ivedimo/Isvedimo
blokas
I/O
Funkcinis blokas
(36)
D/T
Q
18
Ivedimo/Isvedimo
blokas
I/O
Galimas montazinis IR

Pav. 4-10

4.3.5 ĮVEDIMO – IŠVEDIMO BLOKAS



Pav. 4-11

Įvedimo – išvedimo blokas atlieka interfeiso tarp vidinių loginių signalų ir mikroschemos kontaktinių išvadų funkcija pav. 4-11. Kiekviename *įvedimo – išvedimo bloke* yra įvedimo ir išvedimo buferiai, signalų išvedimo leidimo multipleksorius ir programavimo schema vartotojo programuojamam bendrajam išvadui.

Įėjimo buferis gali dirbti su standartiniais CMOS, arba TTL loginiais lygiais veikiančiais nuo 5V arba 3.3 V maitinimo įtampos. Įėjimo buferiai turi atskirą vidinį maitinimą (V_{CCINT} = 5 V) dėl slenkstinio lygio išėjimo signalų stabilumo užtikrinimo nepriklausomai nuo V_{CCIO} įtampos.

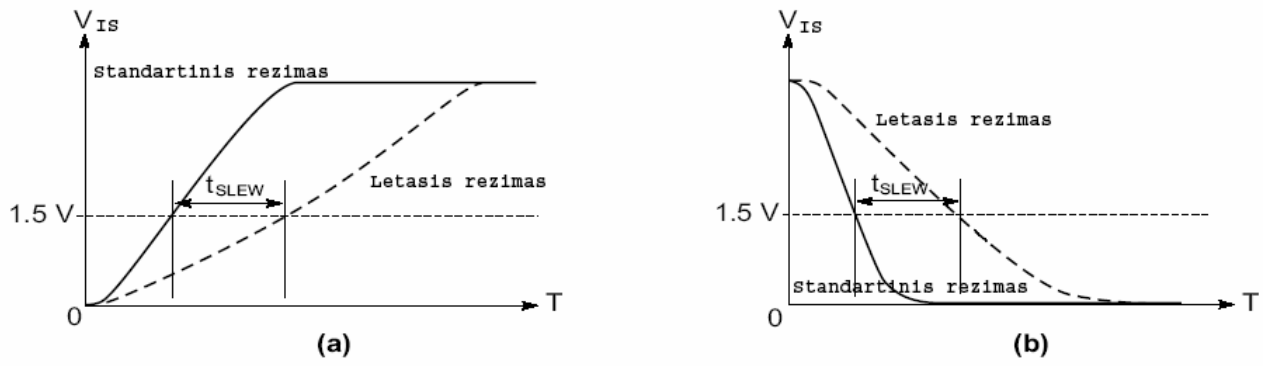
Išėjimo leidimo (OE) signalo funkcijai atlikti gali būti panaudota viena iš termų nuo PTOE makroląstelės išėjimo, arba vienas iš globalių išėjimo leidimo signalų GOE (Global Ouput Enable) teigiamo arba neigiamo poliškumo. Priklausomai nuo makroląstelių kiekio

mikroschemoje (36...144 arba 216...288), tokie globalūs signalai gali būti du arba keturi atitinkamai.

Prie kiekvieno mikroschemos įėjimo/išėjimo tuo metu kai vyksta įrašymas/trynimas, o taip pat maitinimo įtampos įjungimo momentu yra prijungiami vidiniai „pritraukiantieji“ rezistoriai (varža apie 10 kΩ) prie maitinimo įtampos V_{CCINT} (5 V, paduodami į vidinius CPLD blokus) kad būtu išvengta įeinančių/išeinančiu signalų potencialo dreifo, pereinamojo periodo metu.

Pereinant mikroschemai į normalų darbinį rėžimą vidiniai rezistoriai atjungiami.

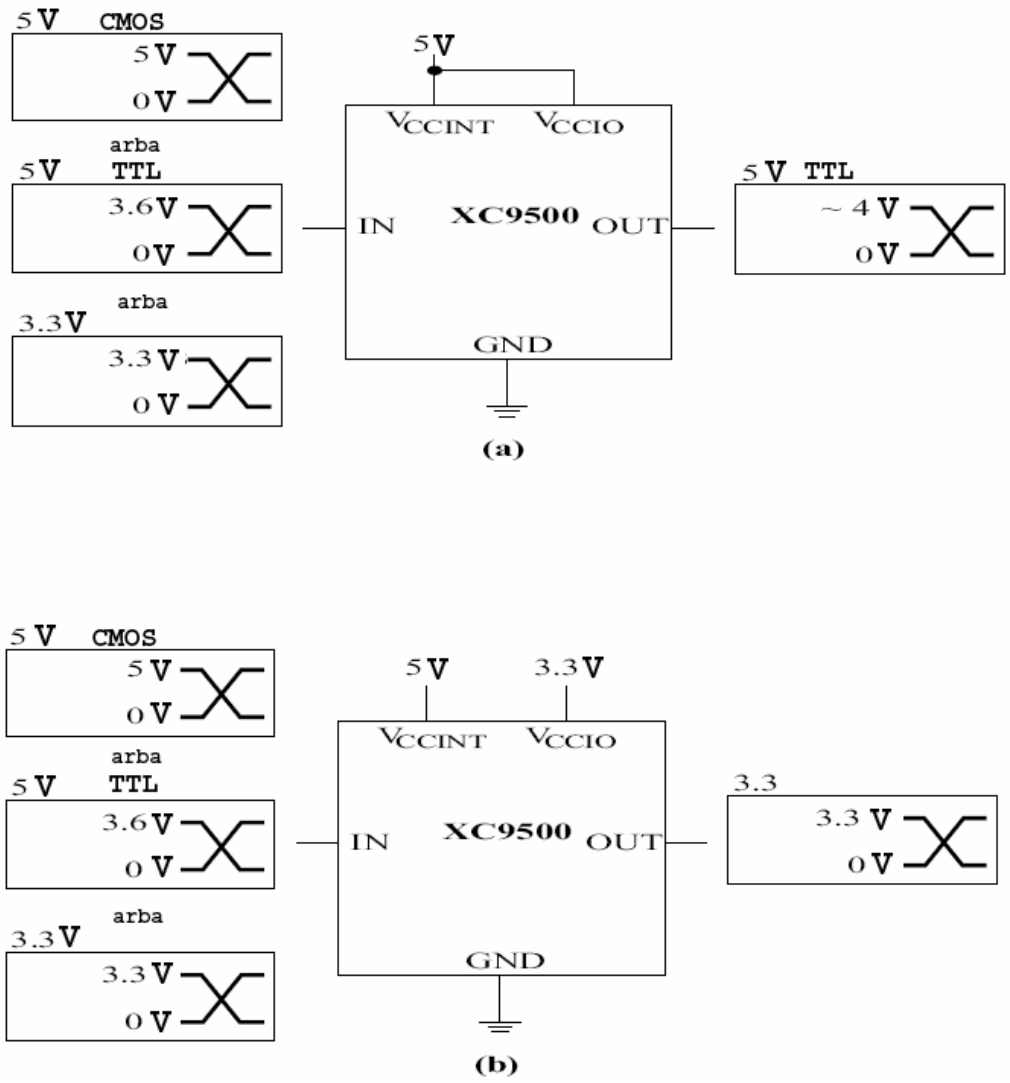
XC9500 mikroschemų šeimoje yra numatyta išėjimo signalo fronto ilgio valdymo galimybė kiekvienam išėjimo buferiui atskirai (Slew Rate), pasinaudojus šia galimybe galima esant reikalui sumažinti triukšmų lygį mikroschemos išėjime, ne didele greitaveikos sumažinimo sąskaita dydžiu t_{SLEW}. *Įvedimo – išvedimo* bloko charakteristikos abiems rėžimams pavaizduotos pav. 4-12.



Pav. 4-12

Siekiant dar didesnio atsparumo triukšmams yra galimybė *įvedimo – išvedimo* bloko viduje bet kuri išėjimo kontaktą prijungti prie bendro (User-Programmable Ground) prievado.

Šios mikroschemų šeimos išėjimo buferiai gali valdyti 24mA srovę, o dėl atskiros maitinimo įtampos (V_{CCIO}) išėjimo grandinėje atsiranda galimybė dirbti su 5V arba 3,3V išėjimo lygiais visuose išėjimuose tuo pat metu. pav. 4-13 pailiustruota XC9500 šeimos mikroschemų panaudojimo galimybė dirbant tu skirtingų maitinimo įtampų įrenginiais (5V/ 3,3V) ir skirtingų technologijų CMOS ir TTL schemomis.

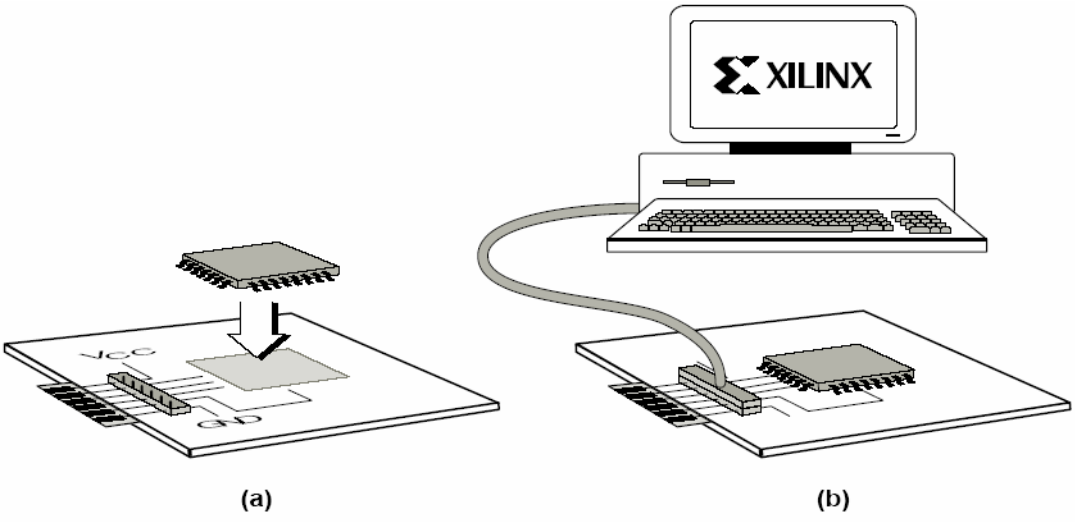


Pav. 4-13

4.3.6 KONTAKTŲ UŽFIKSAVIMO GALIMYBĖ

XC9500 šeimos mikroschemos pasižymi architektūros išskirtinumų, kuris leidžia daryti pakeitimus veikimo schemoje, išsaugant mikroschemos išvadų paskirtį. Tai projektuotojui suteikia pasitikėjimo, kad įėjimo ir išėjimo kontaktų paskirtis išliks nepakitusi esant bet kokiems nenumatytiems veikimo schemos pakeitimams, o tai pagreitina spausdinto montažo plokštės gamybos procesą, nes ją galima pradėti gaminti iš anksto. Prireikus ką nors pakeisti veikimo schemoje kuri reikalauja didesnės loginės talpos lyginant su turima prieš tai pasirinkta mikroschema, yra galimybė pakeisti turima mikroschema į didesnės loginės talpos modelį kuris bus suderinamas pagal kontaktų paskirtį.

4.4 PROGRAMAVIMAS SISTEMOJE



Pav. 4-14

XC9500 šeimos mikroschemos gali būti programuojamuos sistemoje per standartinį keturių kontaktų JTAG interfeisą, kaip pavaizduota pav. 4-14. Programavimas sistemoje leidžia greitai ir efektyviai daryti pakeitimus projekte ir tam nereikia atlituoti ar išiminėti iš lizdo mikroschemos įrenginyje. Firmos Xilinx programinis paketas sukuria konfigūracinę seką, kuri gali būti įrašoma į mikroschemą arba per JTAG jungiamąjį kabelį, arba per projektavimo sistemą palaikančią JTAG protokolą, arba per JTAG įrenginių elektroninį testerį, arba per paprastą mikroprocesorinį interfeisą kuris gali formuoti JTAG komandas.

Minimalus perprogramavimo ciklų skaičius 10000, o įrašytos konfigūracijos išsaugojimo laikas ne trumpesnis kaip 20 metų.

4.5 PERIFERINIO SKANAVIMO PROTOKOLAS IEEE Std. 1149.1

Mikroschemos priklausančios XC9500 šeimai pilnai palaiko periferinio skanavimo protokolą IEEE Std. 1149.1 (JTAG). Palaikomos šios instrukcijų komandos EXTEST, SAMPLE/PRELOAD, BYPASS, USERCODE, INTEST, IDCODE ir HIGHZ. Dėl perprogramuojamumo sistemoje įtrauktos šios papildomos komandos: ISPEN, FERASE, FPGM, FVFY ir ISPEX, kurios yra visiškai suderinamos su 1149.1 standarto papildomų komandų rinkiniu.

TMS, TDI ir TDO kontaktai turi „pritraukiančiuosius“ rezistorius pagal IEEE Std. 1149.1 standartą.

BSDL (Boundary Scan Description Language) failai XC9500 šeimos mikroschemoms yra įtraukti projektavimo programinės įrangos paketą arba juos galima parsisūsti iš Xilinx FTP serverio.

4.6 PROJEKTO APSAUGA NUO KOPIJAVIMO

XC9500 šeimos mikroschemose panaudota pažangi apsaugos nuo nesankcionuoto kopijavimo ar atsitiktinio ištrynimo technologija. Lentelėje 4-3 pateikti keturi apsaugos kodo nustatymo variantai.

	Apsauga nuo skaitymo neįjungta	Apsauga nuo skaitymo įjungta
Apsauga nuo įrašymo neįjungta	Skaitymas leidžiamas Rašymas leidžiamas	Skaitymas draudžiamas Rašymas leidžiamas
Apsauga nuo įrašymo įjungta	Skaitymas leidžiamas Rašymas draudžiamas	Skaitymas draudžiamas Rašymas draudžiamas

Lentelė 4-3 apsaugos kodo nustatymo variantai.

Apsauga nuo skaitymo (saugos kodas) nustatoma vartotojo tam, kad būtų išvengta loginės schemos nuskaitymo ar kopijavimo. Saugos kodas gali būti atšauktas tik visiško mikroschemos ištrynimo atveju. Apsaugos kodas nuo rašymo nustato papildomą apsaugą nuo atsitiktinio mikroschemos ištrynimo ar perprogramavimo pas vartotoją. Vartotojui yra palikta galimybė atšaukti apsauga nuo įrašymo perprogramuojant mikroschemą.

4.7 SUMAŽINTOS ENERGIJOS VARTOJIMO REŽIMAS

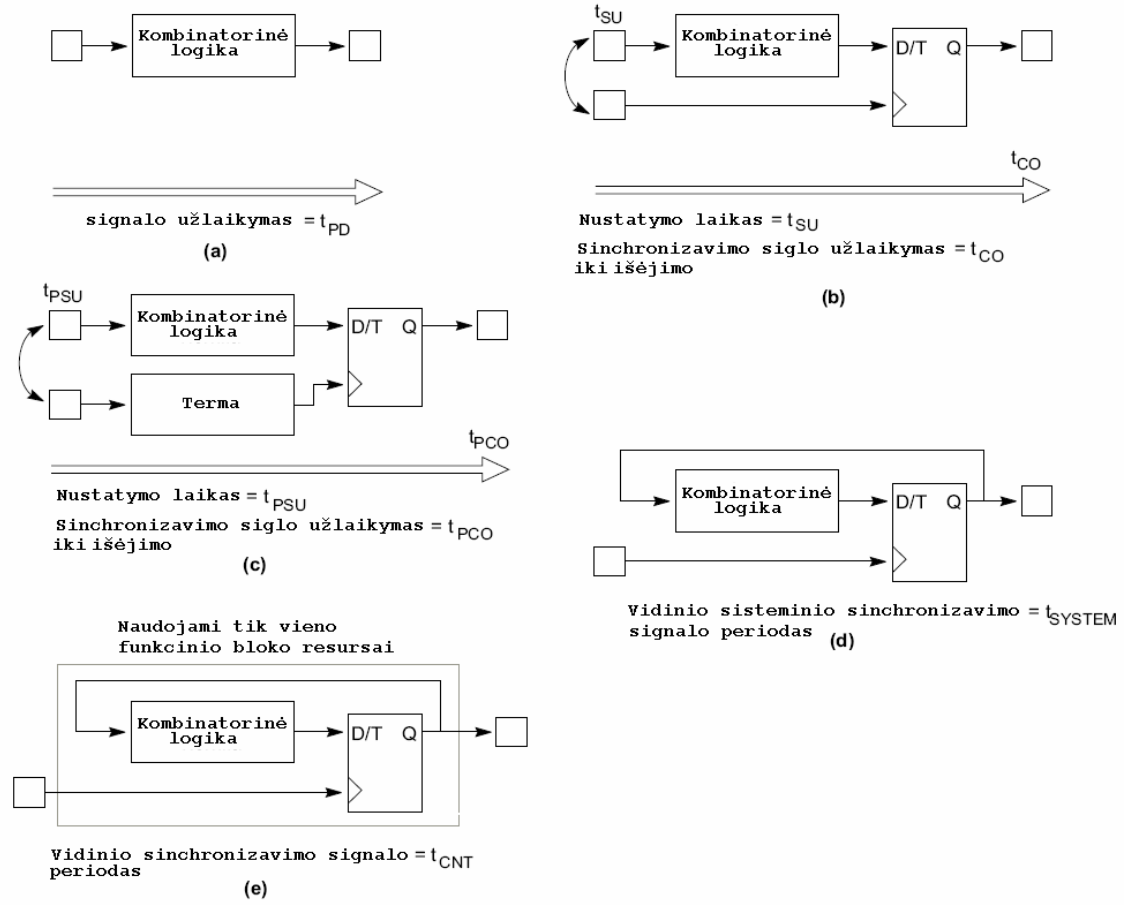
Visose XC9500 šeimos mikroschemose yra numatyta perėjimo į sumažintos energijos vartojimo rėžimą galimybė. Šis rėžimas gali būti nustatomas ne tik visai mikroschemai, bet ir kiekvienai makroląstelei atskirai.

Projektuojant įrenginį reikia nepamiršti, kad sumažintos energijos vartojimo rėžimas įneša papildomą vėlinimą (t_{LP}) makroląstelėje. Tokiu būdu sumažintos energijos vartojimo rėžimą galima įjungti tai loginės schemos daliai kuriai nėra būtina didelė greitaveika, o kur vėlinimo laikas labai svarbus galima nustatyti standartinį energijos vartojimo rėžimą. Ši funkcija leidžia nemažai sutaupyti elektros energijos mikroschemoje.

4.8 SIGNALŲ UŽVĖLINIMO MODELIS

Standartinė XC9500 šeimos mikroschemų architektūra įgalina paprasto signalų vėlinimo modelio panaudojimą visame kristale. pav.4-15 parodytas bazinio modelio pritaikymas

makroląstelei, kurioje panaudotos tik tiesioginės termos esančios standartiniame energijos suvartojimo režime ir su standartiniu frontų užaugimo greičiu. lentelėje 4-4 parodyta kaip įtakojami laikiniai parametrai naudojant ne tik tiesiogines termas, sumažintą energijos vartojimo režimą bei keičiant fronto užaugimo greitį.



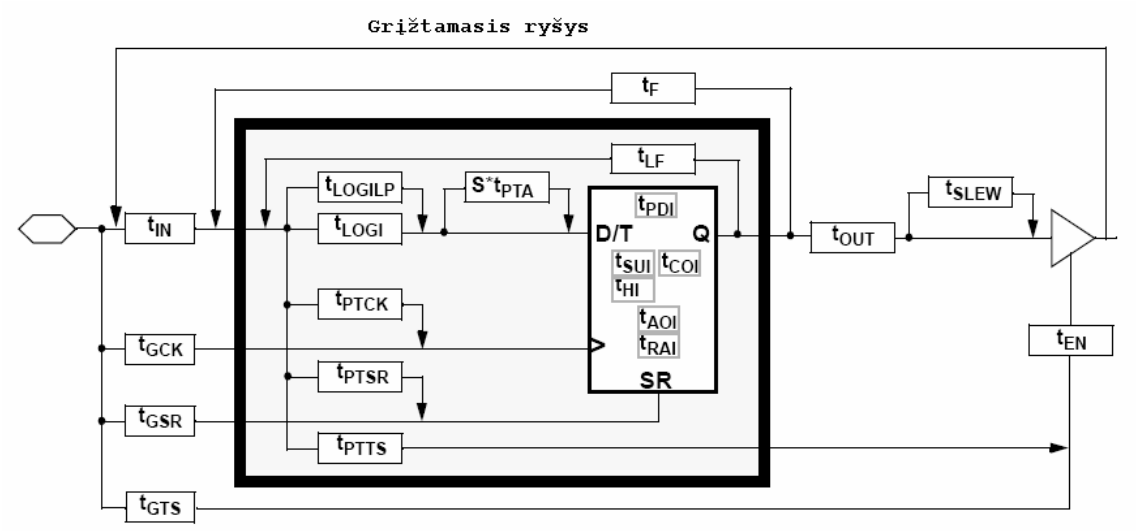
pav. 4-15

Aprašymas	Parametras	Termų skirstytuvas aktyvuotas	Makro ląstelė energijos taupymo rėžime	Fronto užaugimo užlaikymas
Signalo užlaikymas	t_{PD}	+ t_{PTA} * S	+ t_{LP}	+ t_{SLEW}
Globalus sinchronizavimo signalo nustatymo laikas	t_{SU}	+ t_{PTA} * S	+ t_{LP}	–
Globalaus sinchronizavimo signalo užlaikymas iki išėjimo	t_{CO}	–	–	+ t_{SLEW}
Nustatymo laikas naudojant termas	t_{PSU}	+ t_{PTA} * S	+ t_{LP}	–
Globalaus sinchronizavimo signalo užlaikymas iki išėjimo aktyvavus termas	t_{PCO}	–	–	+ t_{SLEW}
Vidinis sistemos periodas	t_{SYSTEM}	+ t_{PTA} * S	+ t_{LP}	–

Lentelė 4-4

Vėlinimas pagal termų pasiskirstymą priklauso koeficiento apimančio logines funkcijas makroląstelėje. Logikos apėmimo koeficientas nustatomas pagal ne tiesioginių termų panaudojimo kiekį minus vienas. Jeigu naudojamos tik tiesioginės termos tai logikos apėmimo koeficientas lygus nuliui. pav.4-7 logikos apėmimo koeficientas lygus vienam, o pav.4-8 2.

Detalus modelis gali būti gaunamas tik iš pilno modelio, parodyto pav.4-16 Parametrų reikšmės ir duomenų paaiškinimą galima rasti konkrečios mikroschemos aprašyme.



pav. 4-16 detalus laikinis modelisw

4.9 MAITINIMO ĮTAMPOS PRIJUNGIMO CHARAKTERISTIKOS

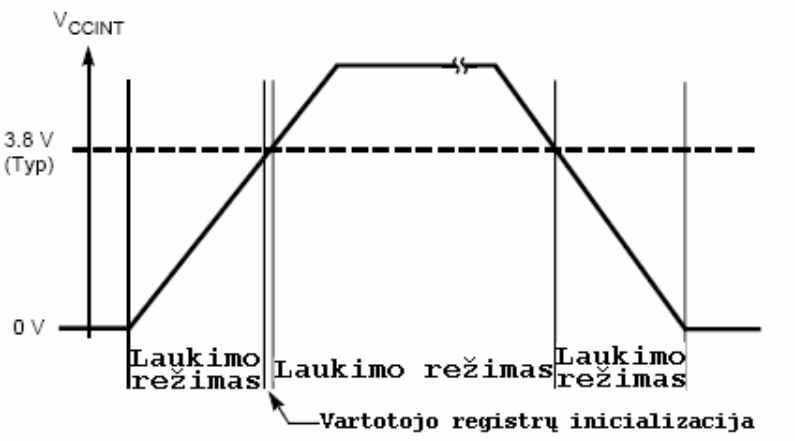
Maitinimo įtampos prijungimo momentu XC9500 šeimos mikroschemose naudojama vidinė schema kuri užlaiko mikroschemą laukimo rėžime iki tol kol maitinimo įtampa V_{CCINT} nepasiekia saugios ribos (apie 3,8 V). Tuo metu visi mikroschemos kontaktai, įskaitant ir JTAG, nepasiekiami kokiems nors poveikiams, „pritraukiantis“ rezistorius aktyvuotas (~ 10 kΩ). žiūrėti į lentelę 4-5. Maitinimo įtampai pasiekus saugią ribą, inicializuojasi naudojami registrai ir mikroschema tuoj pat pereina į normalų darbo rėžimą žiūrėti į pav. 4-17 Inicializacijos laikas 100ms XC9536 –XC95144, 200ms XC95216 ir 300ms XC95288 mikroschemai.

Įrenginio grandys	Laukimo režimas	Įrenginio ištrynimo operacija	Darbinis režimas
„pritraukiantys“ rezistoriai	aktyvus	aktyvus	ne aktyvus
Išėjimai	ne aktyvus	ne aktyvus	kaip užprogramuota
Įėjimai ir sinchronizavimo grandys	ne aktyvus	ne aktyvus	kaip užprogramuota
Funkcinis blokas	ne aktyvus	ne aktyvus	kaip užprogramuota
JTAG-valdiklis	ne aktyvus	aktyvus	aktyvus

Lentelė 4-5

Jeigu mikroschema neužprogramuota, tai visi mikroschemos išvadai, išskyrus JTAG, neprieinami kokiems nors poveikiams, „pritraukiantys“ rezistoriai aktyvuoti (~ 10 kΩ).

Jeigu mikroschema užprogramuota, tai visi mikroschemos išėjimai ir įėjimai, yra atitinkamoje konfigūracijoje, o JTAG prievadai pasiekiami tik programavimo sistemoje ir periferijos skanavimo operacijoms.



pav. 4-17 įrenginio elgsena maitinimo įtampos įjungimo momentu

4.10 PROGRAMINIS PROJEKTAVIMO PALAIKYMAS

Projektavimas palaikomas universaliu Xilinx Foundation Series programiniu paketu. Šis paketas palaiko visus CPLD Xilinx, o taip pat specializuota nemokama programinė įranga WebPack prieinama per internetą (http://www.xilinx.com/sxpresso/webpack.htm).

4.11 GAMYBOS TECHNOLOGIJA

CPLD šeimos mikroschemų gamyboje naudojama patobulinta CMOS technologija - FastFlash. FastFlash technologija buvo specialiai sukurta CPLD architektūrai. Ši technologija pasižymi didele greita veika, greitu programavimu ir palaiko daugiau kaip 10000 įrašymo/trynimo ciklų.

5. VHDL APARATŪROS APRAŠYMO KALBA

VHDL yra skaitmeninės elektroninės aparatūros funkcionavimo aprašymo kalba. Aparatūros aprašymo kalbų sukurta labai daug. VHDL yra aparatūros aprašymo tarptautinis standartas. VHDL leidžia aprašyti bet kokį aparatūriškai realizuojamą algoritmą, todėl VHDL kalba labai panaši į programavimo kalbą. Esminis skirtumas tas, kad VHDL kalba papildomai prie konstantų ir kintamųjų dar turi signalus, su kuriais siejami laiko parametrai, ir galima nurodyti operatorių vėlinimus. Be to, VHDL leidžia aprašyti lygiagrečiai atliekamus procesus.

Tinkamiausia modeliavimo kalba yra VHDL, nes tai universali skaitmeninių elektroninių įrenginių aprašymo kalba. Šia kalba aprašytas įrenginys gali būti modeliuojamas kompiuteryje, o jo veikimas analizuojamas pagal laikines diagramas panašiai kaip su oscilografu. Pagrindinis privalumas tas, kad šia kalba aprašytas įrenginys vėliau gali būti automatiškai sintezuojamas. Automatinės sintezės priemonės užtikrina, kad sintezuota schema veiktų taip pat, kaip ir VHDL aprašytas įrenginys. Todėl elektroninio įrenginio VHDL aprašas pakeičia maketą, ir projektas gali būti demonstruojamas VHDL modeliavimo priemonėmis. Šiuo metu jau yra sukurtų programinių priemonių, leidžiančių iki galo modeliuoti ir gauti bet kokio VHDL kalbos aprašo laikines diagramas.

Laiko parametrai įgalina vartoti VHDL kalbą aprašyti ne tik aparatūrą, bet ir kitas realaus laiko sistemas. VHDL gali būti vartojama kaip imitacinio modeliavimo kalba sudėtingoms sistemoms, aprašomoms stochastinėmis priklausomybėmis. VHDL gali būti ir vykdomoji specifikavimo kalba. VHDL kalba galima modeliuoti sistemas, aprašomas diferencialinėmis lygtimis. Ir visa tai palaiko grafinė įėjimo poveikių įvedimo ir rezultatų išvedimo sąsaja.

Didėjant skaitmeninių sistemų apimčiai ir sudėtingumui, vis plačiau aparatūros projektavimo procese vartojamos kompiuterinės priemonės. Vienas svarbesnių projektavimo metodologijos papildymų buvo aparatūros aprašymo kalbų atsiradimas. Bazuojant šiomis kalbomis, sukurtos įvairios projektavimo sistemos, kurios plačiai vartojamos, projektuojant naują aparatūrą.

Tipinis skaitmeninės sistemos projektavimo procesas parodytas 1.1 pav. Projektavimo procesas prasideda nuo idėjos. Išsamesnis planuojamos aparatūros aprašymas turi būti išplėtotas iš pradinės projekto idėjos. Todėl projektuotojas turi sudaryti projektuojamos sistemos elgsenos aprašymą. Šio projekto etapo rezultatas gali būti duomenų srautų diagrama, duomenų srautų grafas ar pseudokodas. Šis aprašymo lygmuo yra abstrakčiausias. Jis aprašo projekto funkcijas, bet nenurodo, kaip jos turėtų būti įgyvendintos. Funkcijos tik susieja projekto išėjimus su projekto įėjimais. Šio lygmens aprašymas gali būti vartojamas kaip projekto dokumentacija ir suprantamas ne specialistui.

6. JTAG PRAKTINIS TAIKYMAS

Ar jūs kada nors norėjote suprojektuoti sudėtingą loginį įrenginį – be mikroprocesoriaus, bet tikrą skaitmeninę logiką? Tikrai, mikroprocesoriai leidžia programiškai realizuoti daug sudėtingų loginių funkcijų, bet kartais nėra realių techninės įrangos pakaitų.

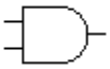
Pavyzdžiui, tarkime, kad jūs turite kontroliuoti 16 įėjimų vienu metu? Mikroprocesorius realiai negali visko sekti vienu metu ir vykdyti kitas užduotis. Jis nuskaitys įėjimus – galbūt labai greitai – bet ten gali būti vėlinimas tarp signalo pasirodymo ir programos atsako. Jūs galite naudoti pertraukimus, padedančius sumažinti šią problemą, bet tik dalis techninės įrangos gali realizuoti lygiagrečių procesų modeliavimą nesiimant analogiško apdorojimo.

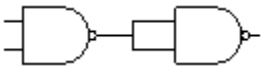
Kad būtų pasiektas darbo tikslas, pakalbėkime apie programuojamos logikos įrenginius (PLDs) kaip bendrą ICs kategoriją, kuri gali būti vartotojo programuojama įvairioms funkcijoms. Bet vis dėlto, programinė įranga ICs nėra tapati mikroprocesoriams. Praktiškai, yra keletas PLDs tipų įskaitant CPLDs (kompleksinės PLDs) ir FPGAs (programuojamų loginių ventilių matrica). Visi šie lustai yra iš esmės vienodi, svarbiausias skirtumas yra galimybė imituoti sudėtingą schemą ir kaina.

Pagrindinė idėja yra paprasta:

- Jūs apibūdinate projektą, kurį norite realizuoti naudojant principinę schemą arba programavimą, kaip kalbą. Daugeliu atvejų naudojame abu šiuos metodus viename projekte.
- PLD gamintojų siūlomi įrankiai leidžia imituoti projektą jūsų kompiuteryje. Jei tai veikia imitavimo režime, jūs turite didelę tikimybę, kad šis projektas realiai veiks.
- Tada šie įrankiai kompiliuoja jūsų projektą į bitų seką, kurią galite įrašyti į lustą. Taip yra gaunamas pritaikytas individualiam vartotojui lustas. Modernūs PLDs gali būti perprogramuojami daugelį kartų.

Svarbu suprasti, kad šios bitų sekos tai nėra programos, iš esmės tai nuoseklių instrukcijų sąrašas. PLDs savyje talpina šimtus arba tūkstančius loginių ventilių ir trigerių. Jie taip pat aprūpinti milžinišku kiekiu programuojamų perjungiklių. Darbo kompiuteryje programinė įranga suvokia, kaip sujungti vidinius įėjimus - išėjimus, kad jie atitiktų loginį jūsų projekto schemos aprašymą. Programa nustato sujungimus tarp vidinių įėjimų-išėjimų.

Žinoma, galutinis rezultatas gali būti nepanašus į mūsų sugalvotą, bet turi atlikti tą pačią funkciją. Hipotetinis pavyzdys, tarkime jūs nusprendėte suprojektuoti IR elementą. Tačiau jūsų PLD sudarytas tik iš IR-NE elementų. Mūsų projektas gali atrodyti taip : 

Bet galutinė realizacija gali atrodyti šitaip: 

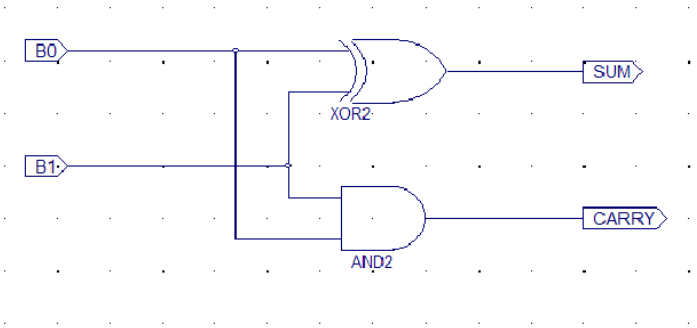
Programinė įranga supranta, kaip padaryti IR elementą iš IR-NE elemento panaudojant dar vieną IR-NE elementą kaip invertorių. Žinoma, tai paprastas pavyzdys. CPLD gali modeliuoti keletą tūkstančių loginių elementų o kai kurios FPGAs daug didesnės ir už ją. Turint pakankamai didelį lustą įmanoma sukurti net procesorių!

6.1 PRINCIPINĖS SCHEMOS METODAS

Taigi PLDs yra nuostabi, ar ne? Kodėl ne kiekvienas ją naudoja? Yra keletas kliūčių kurias reikia įveikti. Pirma, dauguma PLDs yra korpusuose su kuriais sunku dirbti. Antra, iki šių dienų, daugumai PLDs modeliavimui reikia brangios techninės ir programinės įrangos.

Laimei, galime šias kliūtis įveikti turėdami pigių prie personalinio kompiuterio jungiamą programatorių, kuris leidžia lengvai dirbti su keletu Xilinx PLDs (žr. pav. 4-1). Šis programatorius leidžia mums programuoti PLD per personalinio kompiuterio lygiagretų prievadą (pakanka turėti tik vieną programatorių). Plokštė pateikta (priede.....) gali būti maketu panaudojant prilituojamus jungiamuosius laidus. Galų gale galime parsiųsti nemokamą programinę įrangą iš Xilinx Web puslapio http://www.xilinx.com/sxpresso/webpack.htm.

Šis darbas supažindina su modeliavimu, panaudojant Xilinx programinę įrangą ir pagamintą maketinę plokštę (žr.priede nr.....) Paanalizuokime, kaip paruošti pussumatoriaus projektą.

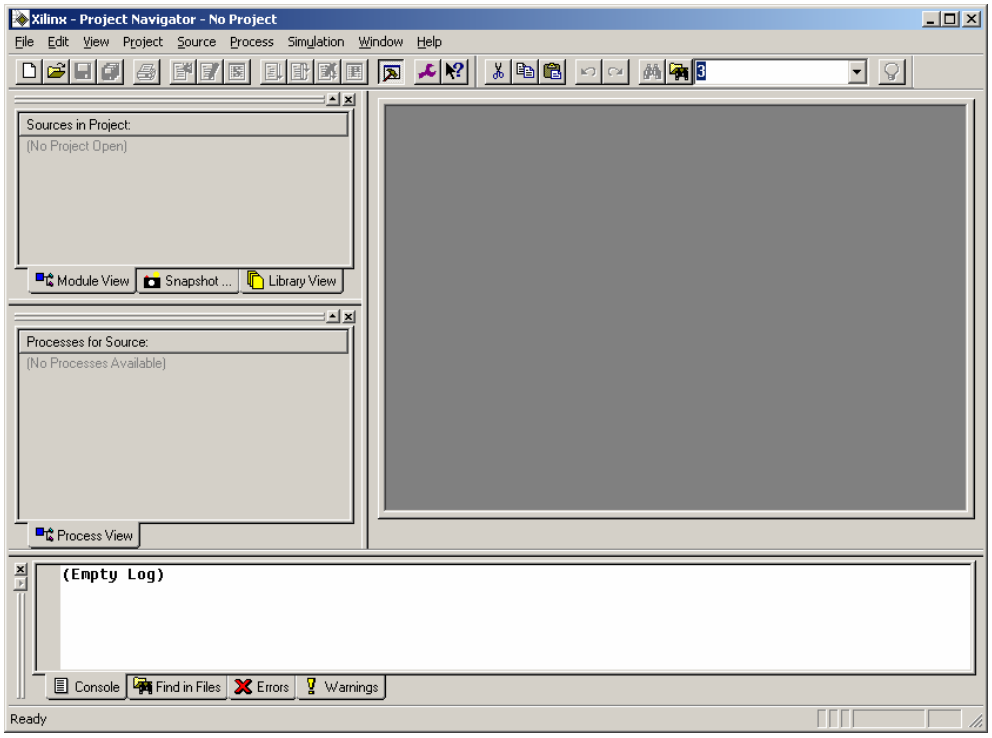


pav. 6-1

Šioje paprastoje schemoje naudojami du B0 ir B1 loginiai įėjimai, kurie sudedami ir pateikiami kaip rezultatai loginiuose SUM ir CARRY išėjimuose.

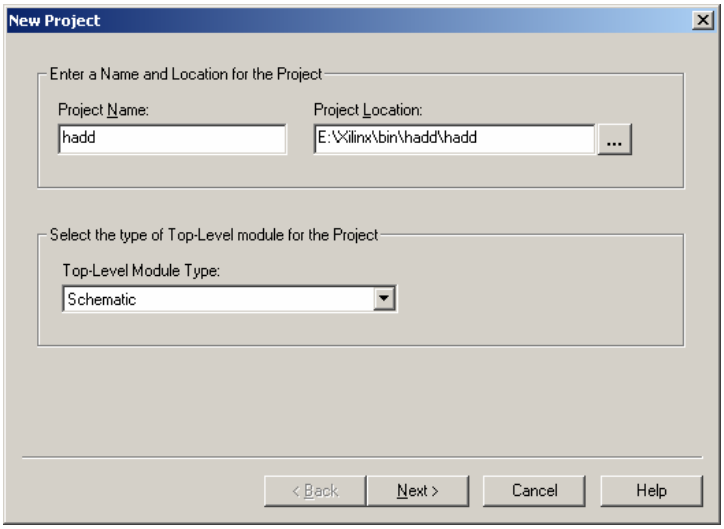
Pradžioje mums reikia instaliuoti Xilinx Web Pack programinę įrangą į personalinį kompiuterį. Nuo tada ji gali imituoti mūsų projektą. Mums nereikia jokios techninės įrangos, norint pamatyti rezultatus. Bet jeigu mes norime realizuoti tikrą techninę įrangą mums prireiks Xilinx XC95108 PLD, maketinės plokštės, ir pasigaminto programatoriaus.

Pradėkime pussumatoriaus projektą:



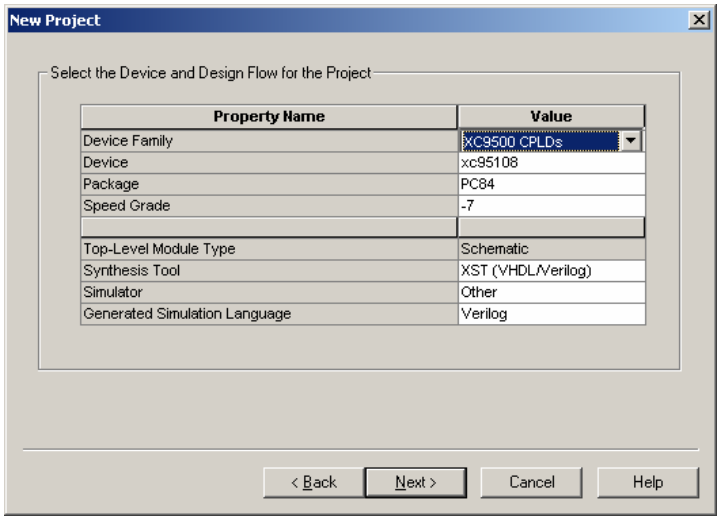
Pav. 6-2

Paleidę WebPack Project Menedžerį pamatysime tuščią programos darbo langą pav. 6-2. Sukurkime naują projektą (File|NewProject...). Pamatysime šį dialogą pav. 6-3:

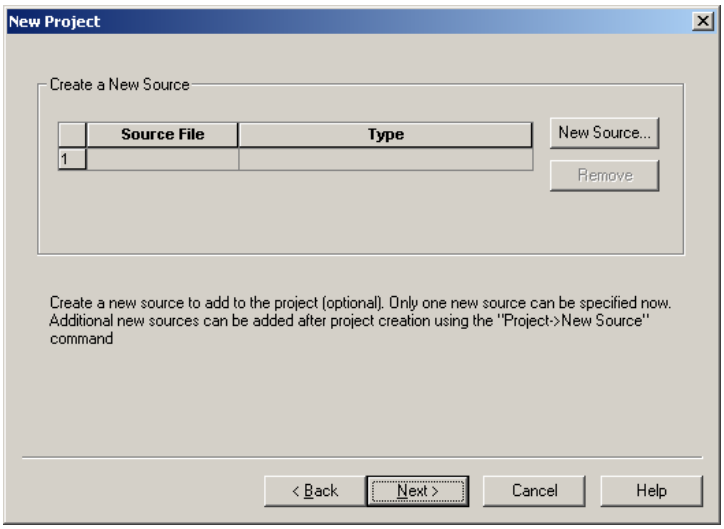


Pav. 6-3

Pavadinkime projektą hadd. Projekto dislokacijos vieta diske nustatoma automatiškai, bet jei norime ją galime pakeisti. Šiam projektui mums reikia XC 95108 mikroschemos ir pasirinkime XST Verilog (taip pat šiam paprastam projektui, EDIF irgi tiks) projekto tipą. Paspauskime Next.

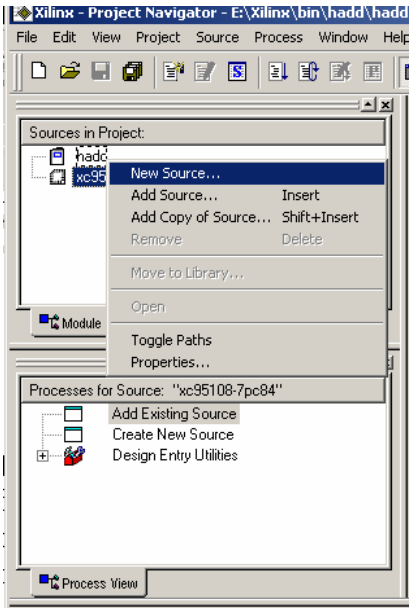


Pav. 6-4



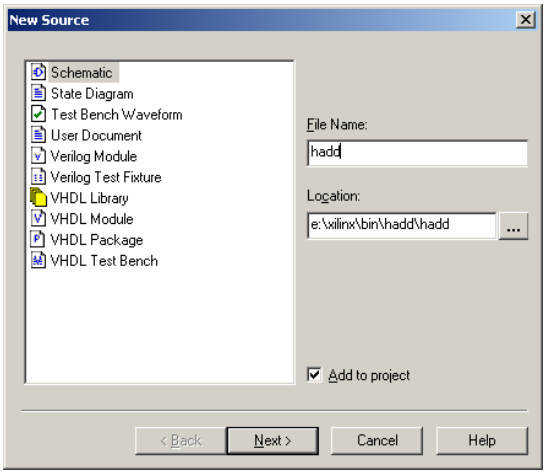
Pav. 6-5

Toliau kairėje ekrano pusėje pamatysime keletą langų. Viršutinis langas parodys projekto failus. Žemesniame lange galėsime pamatyti skirtingas operacijas, kurias galime vykdyti pasirinktam projekto failui. Šiam projektui mes norime nubrėžti schemą. Dešiniu pelės klavišo paspaudimu ant hadd pamatysime meniu langą pavaizduotą pav. 6-6. Pasirinkime New Source.

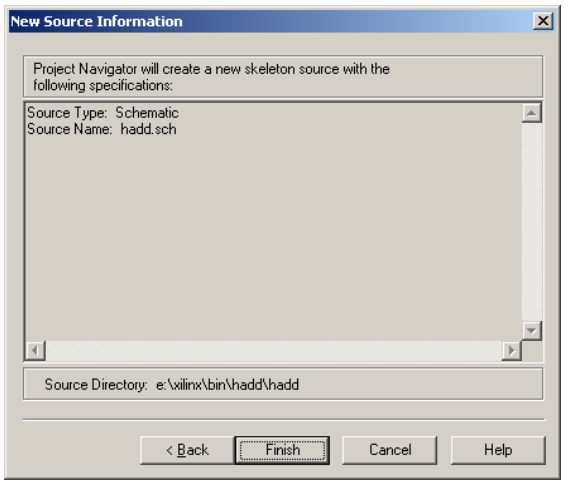


Pav. 6-6

Taip yra iškviečiamas dialogas, kur galime pasirinkti norimo sukurti failo tipą. Pasirinkime dokumento tipą „Schematic“. Failą pavadinome hadd, nuo to laiko tai bus pagrindinis projekto failas. Paspauskime Next. Pamatysime patvirtinantį pranešimą, tada tiesiog paspauskime Finish.

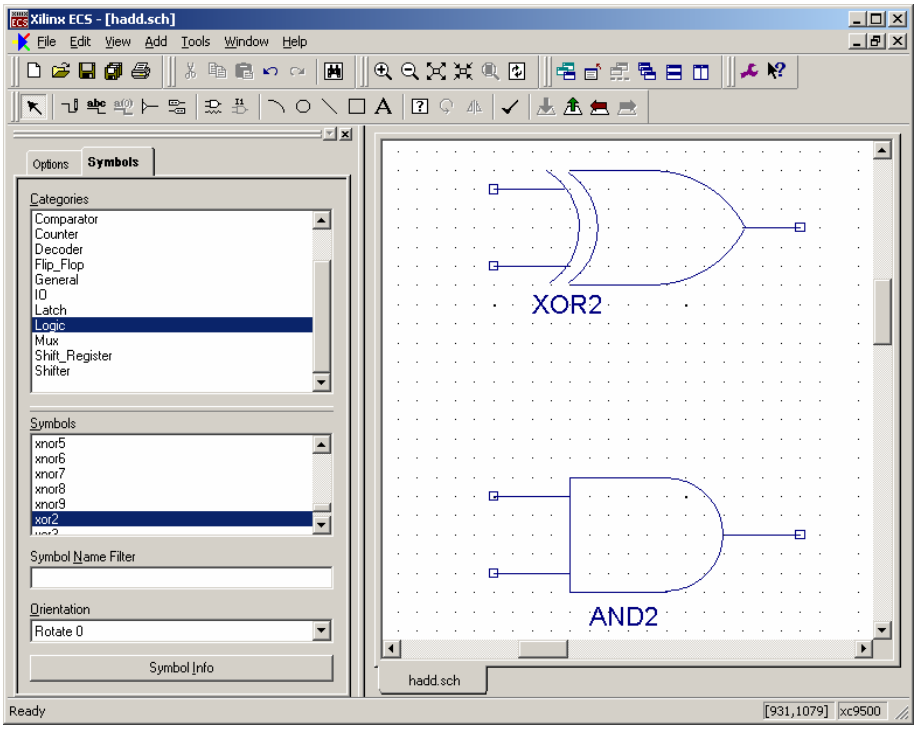


Pav. 6-7

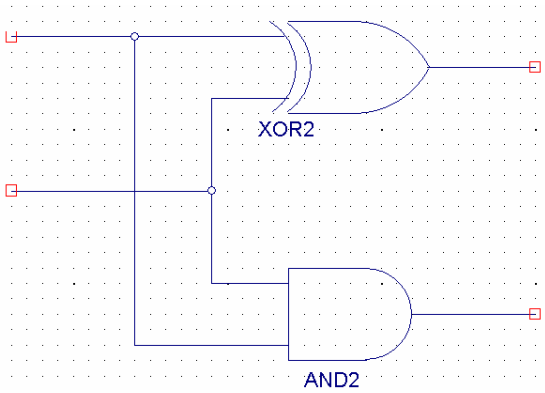


Pav. 6-8

Tai paleidžią kitą programą, scheminį redaktorių. Dabar galime pradėti braižyti schemą. Kairėje scheminio redaktoriaus pusėje pamatysime langą, sudarytą iš kategorijų ir simbolių. Reikiami simboliai yra Logic kategorijoje ir pavadinti xor2 ir and2 (du įėjimai xor ir and elementas su dviem įėjimais). Pasirinkime kiekvieną elementą ir nutempkime pele į schemos lauką, elemento pritvirtinimui paspauskime kairį pelės klavišą. Tada galime panaudoti Add Wire komandą (arba Add Wire įrankių mygtuką esantį netoli lango viršaus) sujungimų piešimui. Galutinis rezultatas turėtų būti panašus į pavaizduotą pav. 0-9.



Pav. 6-9



Pav. 6-10

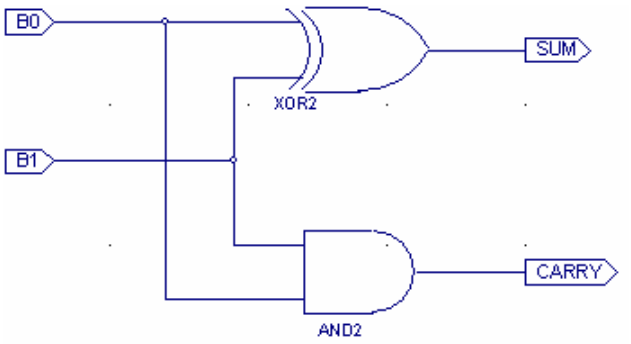
Atkreipkime dėmesį į tai kad elementų išėjimai turi trumpus jungiamuosius laidus. Tai būtina padaryti, nes negalima tiesiogiai prie elemento išėjimo jungti I/O žymeklio. Prijunkime I/O žymeklius prie elementų jungiamųjų laidų pratęsimų.

Nupiešti jungiamąjį laidą galima dvigubu pelės spragtelėjimu jo gale net neišsirenkant jungiamojo laido piešimo įrankio. Tai specialus palengvinimas, kuris padeda piešti jungiamuosius laidus bet kur.

Taip pat galima rasti papildomos išsamios Xilix`s dokumentacijos (kuri nepateikiama kartu su WebPack paketu) adresu http://toolbox.xilinx.com/docsan/xilinx4/manuals.htm

Pasirinkime I/O žymeklio pridėjimo komandą iš meniu (arba iš įrankių juostos, arba paspauskime Ctrl+G klavišų kombinaciją). Netoli ekrano viršaus pamatysime tris atsiradusius alternatyvių veiksmų pasirinkimo mygtukus: Input (įėjimas), Output (išėjimas) ir Bidirectional (dvikryptis). Pasirinkime Input ir spragtelėkime pele ant abiejų įėjimų. Spragtelti reikia tiksliai ant jungiamojo laido galo, kitu atveju I/O žymeklis nepasidės, o šią operaciją turėsime pakartoti iš naujo. Persijunkime į Output ir uždėkime du išėjimų žymeklius. Prisiminkime, kad negalima tiesiai prie elemento įėjimo/išėjimo prijungti I/O žymeklio, taigi reikia elementų įėjimus ir išėjimus pratęsti jungiamuoju laidu, nes tik tada bus galima prijungti I/O žymeklį.

Pridėtiems žymekliams bus automatiškai sugeneruojami pavadinimai, kurie neturi jokios prasmės. Dvigubu pelės spragtelėjimu ant žymeklio galima pakeisti jo pavadinimą. Pakeiskime pavadinimus atitinkamai pav. 0-10.



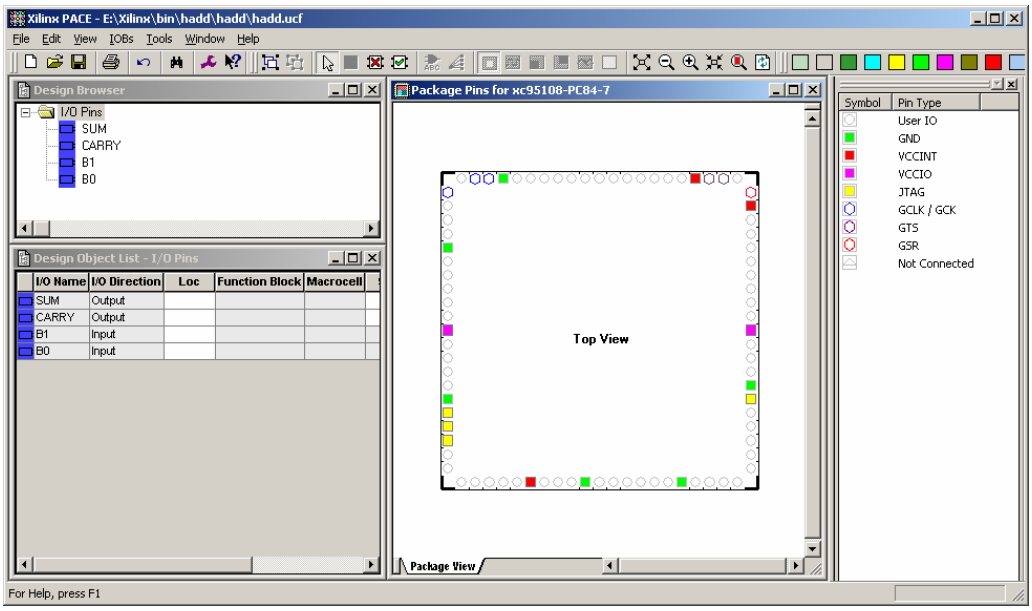
Pav. 6-11

Ir viskas! Prieš paliekant šią programą, paspauskime Tools|Check Schematic. Taip patikrinsime projektą. Turėtumėm gauti pranešimą, kad klaidų nėra. Jei vis dėlto gausime pranešimą apie klaidą pataisykime ją prieš tęsdami. Dažniausiai pasitaikančios klaidos - tai jungiamųjų laidų sujungimo klaidos.

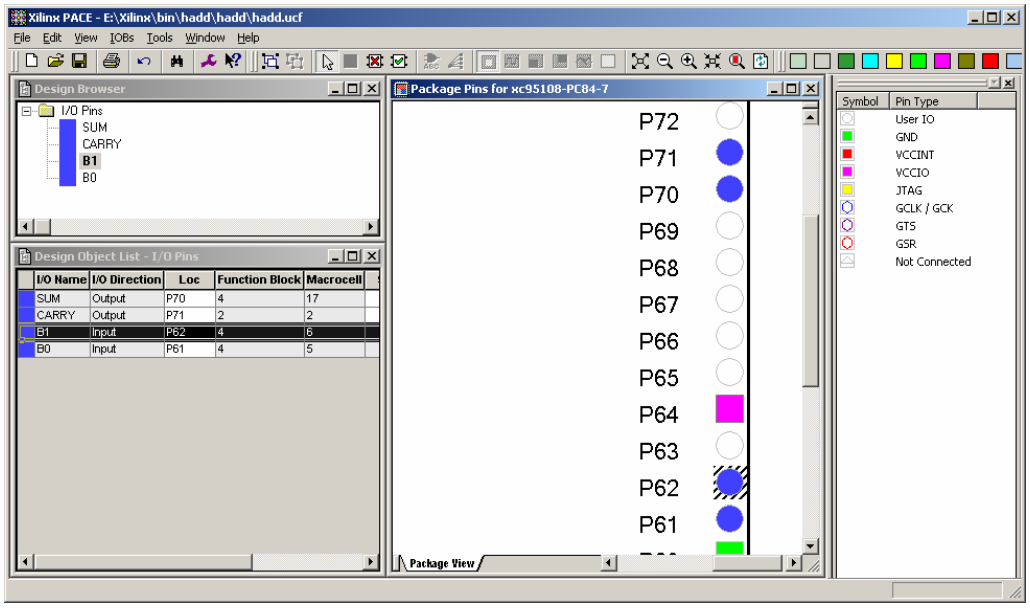
Persijunkime atgal į Project Navigator programą. Mums būtina I/O žymeklius priskirti realiems lusto išvadams. Jei mums tai nerūpi galima praleisti šį žingsnį tada kompiliatorius pats priskirs I/O žymeklius realiems išvadams. Bet dažniausiai mes turime schemą, kurioje nurodyta, kurie išvadai kokius I/O žymeklius turi atitikti (fiksuotai).

Šiame projekte panaudokime 70 ir 71 išvadus SUM ir CARRY, o B0 ir B1 61 ir 62. Įsitikinkime, kad hadd (hadd.sch) pažymėtas Sources projekto lange. Tada žemiau esančiame lange

spragtelėkime pliuso ženklą esantį šalia Design Entry Utilities. Po to spragtelkime pliuso ženklą ties User Constraints. Dvigubai spragtelkime Assign Package Pins. Atsidarys Xilinx PACE langas pav. 0-11

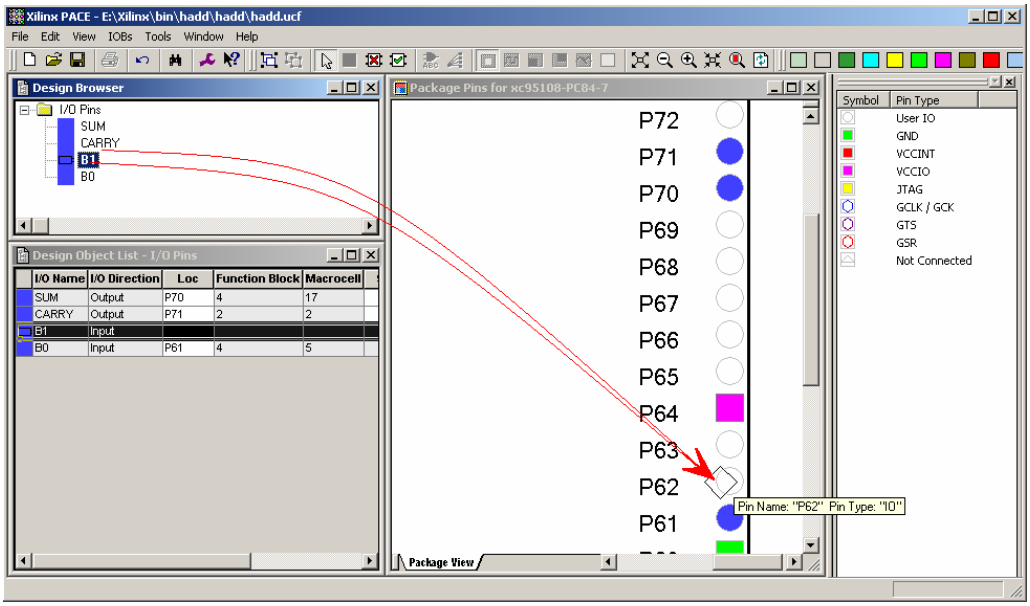


Pav. 6-12



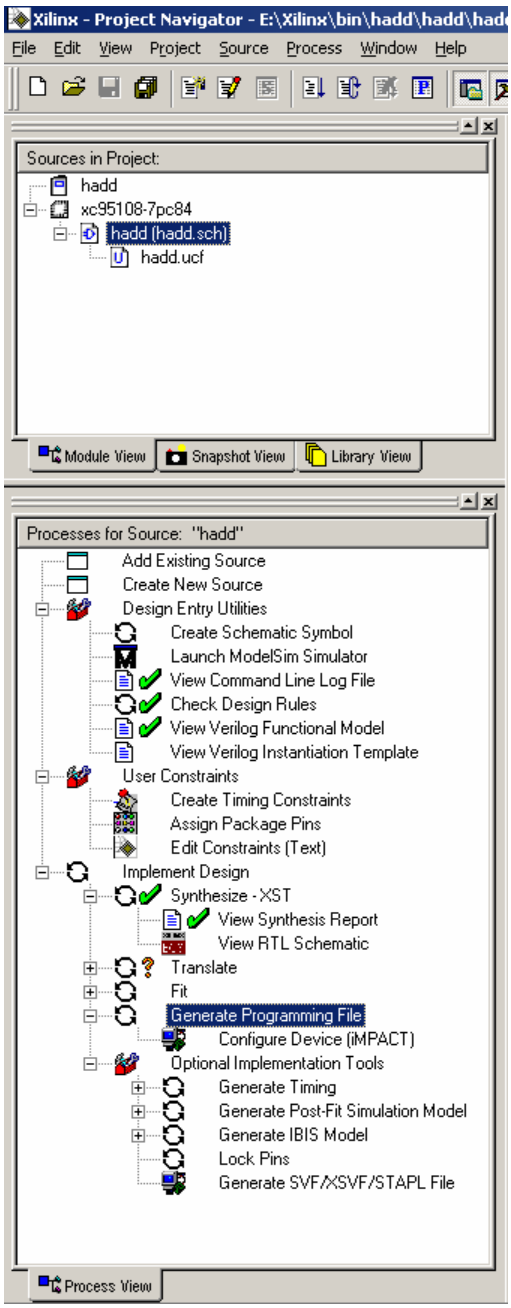
Pav. 6-13

Pasinaudokime + įrankiu ir padidinkime lusto korpuso brėžinį kol pamatysime išvadų numeraciją. Tada tiesiog nutepmkime pele iš Design Browser lango I/O žymeklį ant norimo išvado.



Pav. 6-14

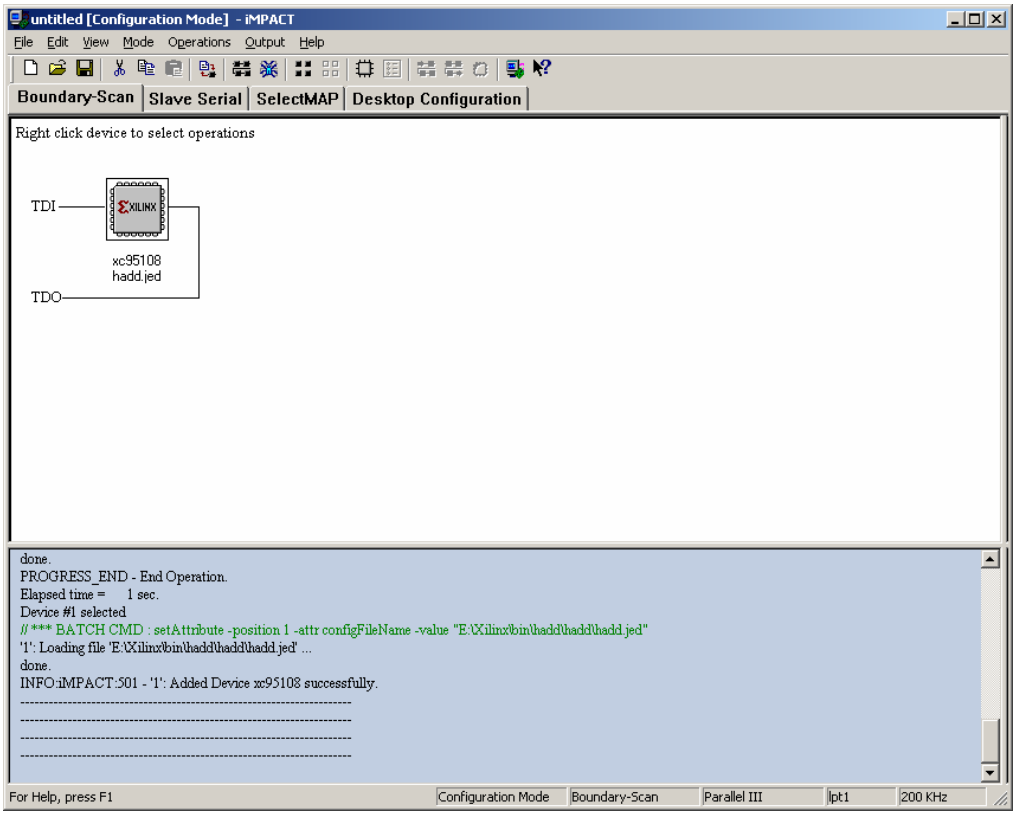
Uždarykime Xilinx PACE programą išsaugodami pakeitimus ir vėl sugrįžkime į Project Navigator. Sekantis žingsnis bus projekto kompiliavimas ir lusto programavimas. Jei projektas būtų žymiai sudėtingesnis reikėtu pirma jį simuliuoti (imituoti), bet dabar praleiskime šį žingsnį.



Pav. 6-15

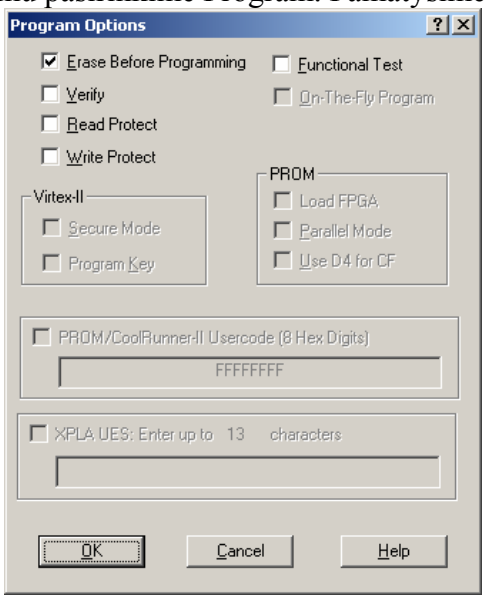
Dukart spragtelėkime Configure Device (iMPACT) ir kompiliatorius pats atliks visus praleistus žingsnius ir paleis iMPACT programą mūsų lusto programavimui. iMPACT programos dokumentaciją galima rasti internete, bet kol tik programuojame lustą, mums, turbūt, jos niekada neprireiks.

Mes jau turime būti prijungę JTAG programatorių prie lygiagretaus personalinio kompiuterio prievado ir prisijungę prie maketo. Ir abiems šioms plokštėms turi būti prijungtas maitinimo šaltinis. iMPACT programa turi aptikti mūsų JTAG kabelį ir grafiškai pavaizduoti programuojamą lustą.



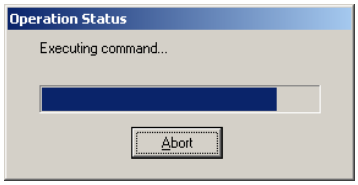
Pav. 6-16

Dešiniu pelės klavišo paspaudimu pasirinkime Program. Pamatysime šį dialogą:



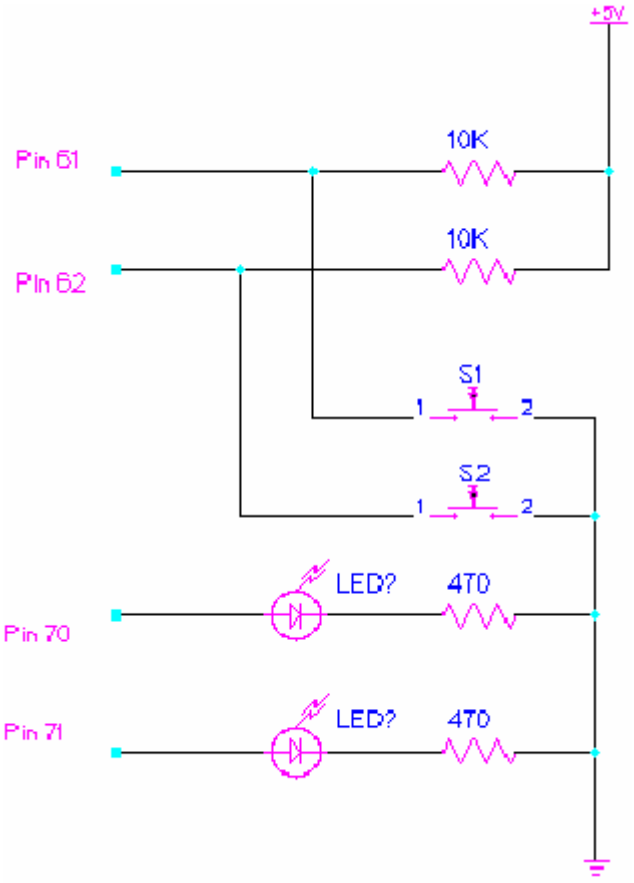
Pav. 6-17

Tiesiog paspauskime OK. Kol indikatorius parodys progresą reikės palaukti apie 15-30 sekundžių:



Pav. 6-18

Kai progresas pradeda vykti tada operacija greitai baigiama. Lustas pradeda veikti iš karto jį užprogramavus. Atitinkamai prijunkime 70 ir 71 lusto kojeles per rezistorius (325 omų) prie šviesos diodų. Taip pat prijunkime 61 ir 62 lusto kojeles per šuntuojančius rezistorius ir jungtukų pagalba galėsime jas nustatyti į nulinį lygį. Pagal schemą pav. 0-18.



Pav. 6-19

Turime galėti manipuliuoti jungikliais ir pamatyti tai:

0 + 0 = 0 0

0 + 1 = 0 1

1 + 0 = 0 1

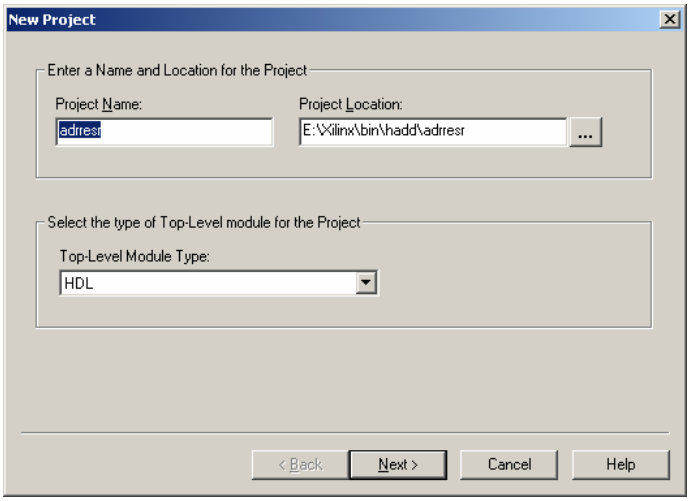
1 + 1 = 1 0

Sveikiname! Mes užprogramavome CPLD!

6.2 VHDL PROGRAMAVIMO KALBOS METODAS

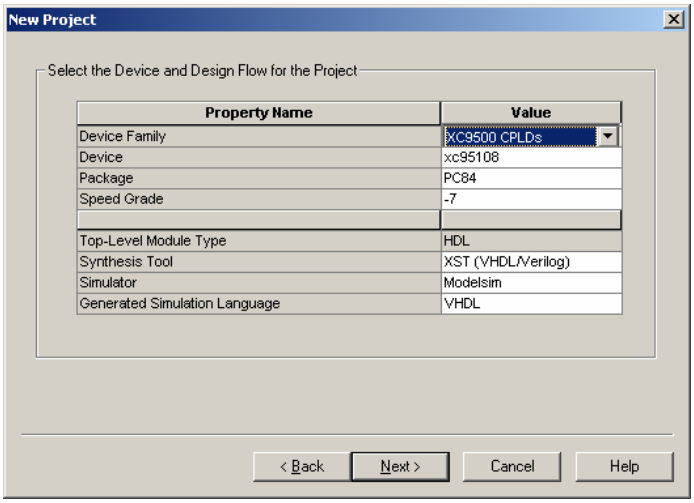
8 bitų adreso atpažinimo įrenginio projektas:

Paleidę WebPack Project Menedžerį, pamatysime tuščią programos darbo langą pav. 6-2. Sukurkime naują projektą (File|NewProject...). Pamatysime šį dialogą pav. 6-20



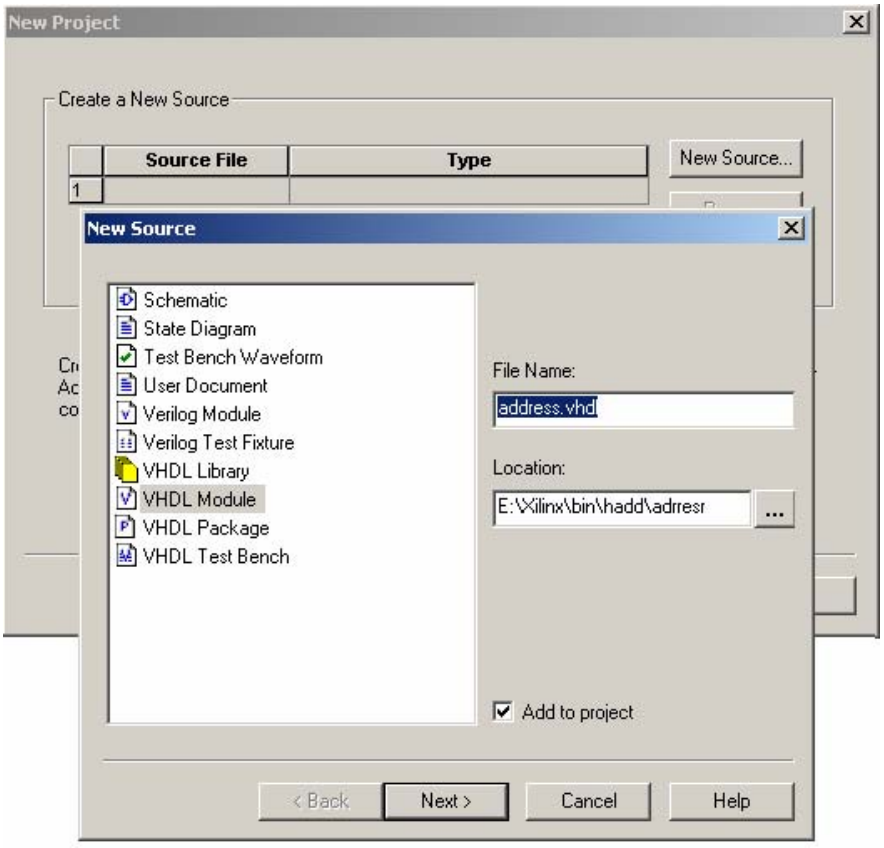
pav. 6-20

Pavadinkime projektą adrresr. Projekto dislokacijos vieta diske nustatoma automatiškai, bet jei norime ją galime pakeisti. Šiam projektui mums reikės XC 95108 mikroschemos ir pasirinkti HDL projekto tipą. Paspauskime Next.



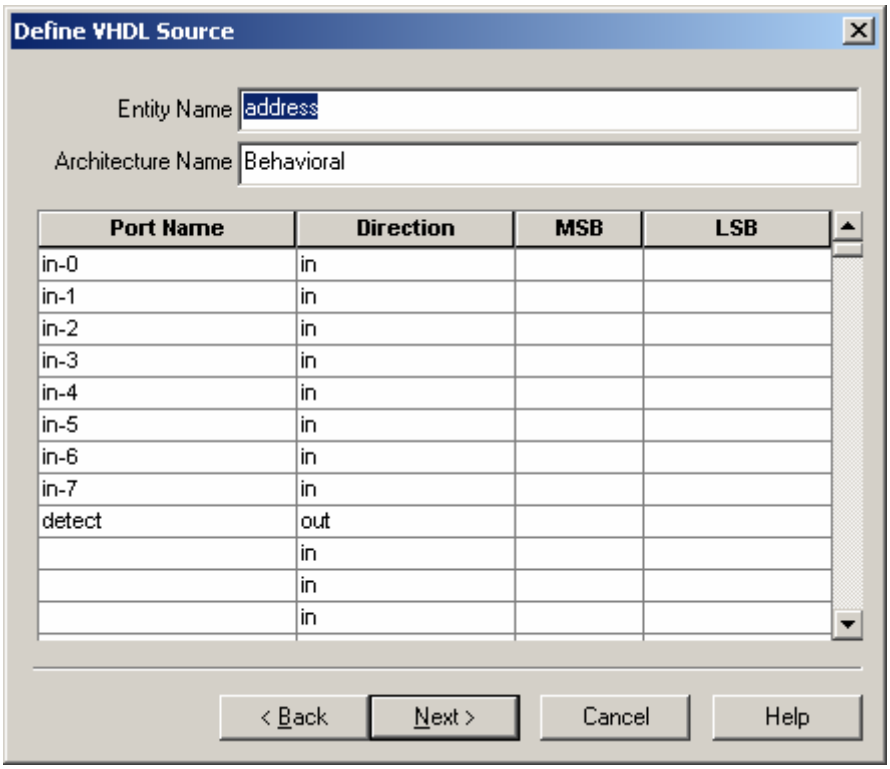
pav. 6-21

Šiame projekte mes norime naudoti VHDL programavimo kalbą. Kairiu pelės klavišo paspaudimu ant New Source... mygtuko pamatysime meniu langą pavaizduotą pav. 6-22. Pasirinkime VHDL Module ir įrašykime address.vhdl bylos vardą. Paspauskime Next.



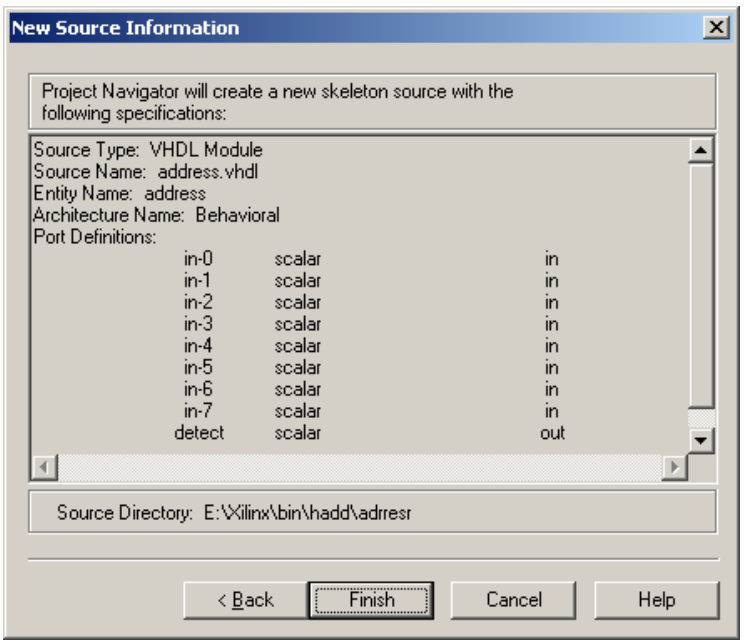
pav. 6-22

Toliau programa paprašys nurodyti projekte naudojamų išvadų kryptis bei jų pavadinimus.

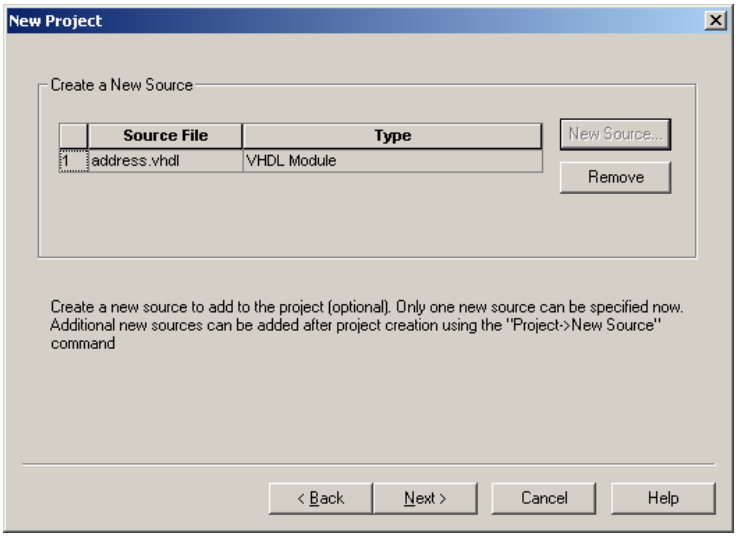


pav. 6-23

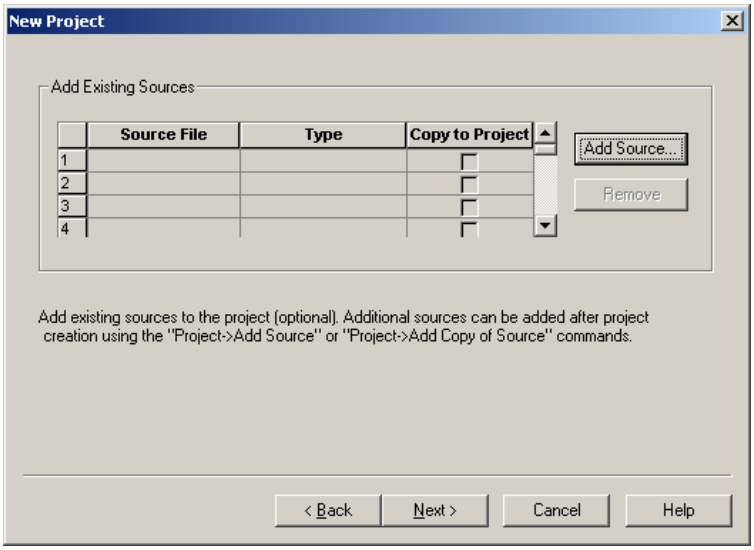
Paspauskime Next ir perėję prie lango pav. 6-24 Finish.



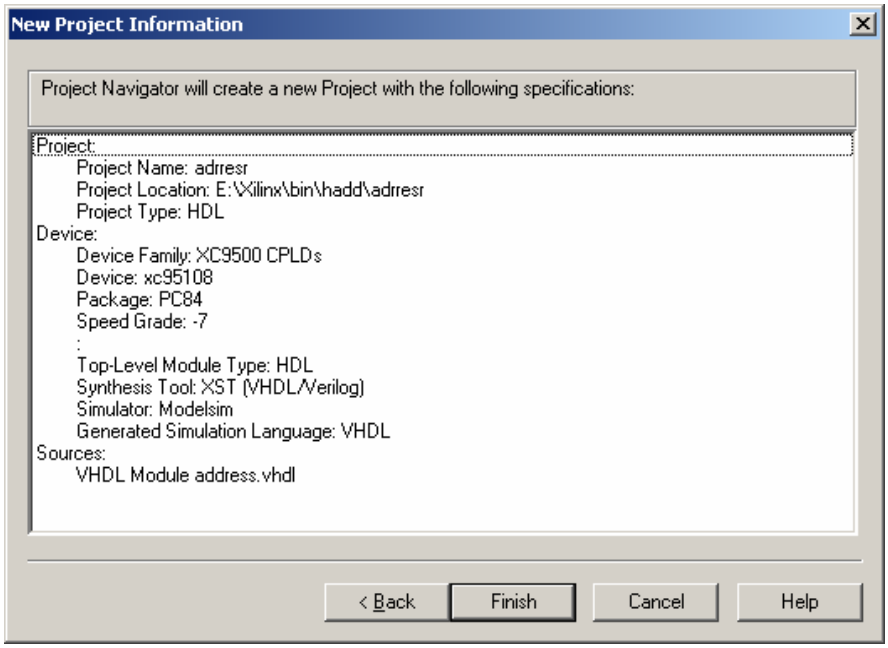
pav. 6-24



pav. 6-25



pav. 6-26

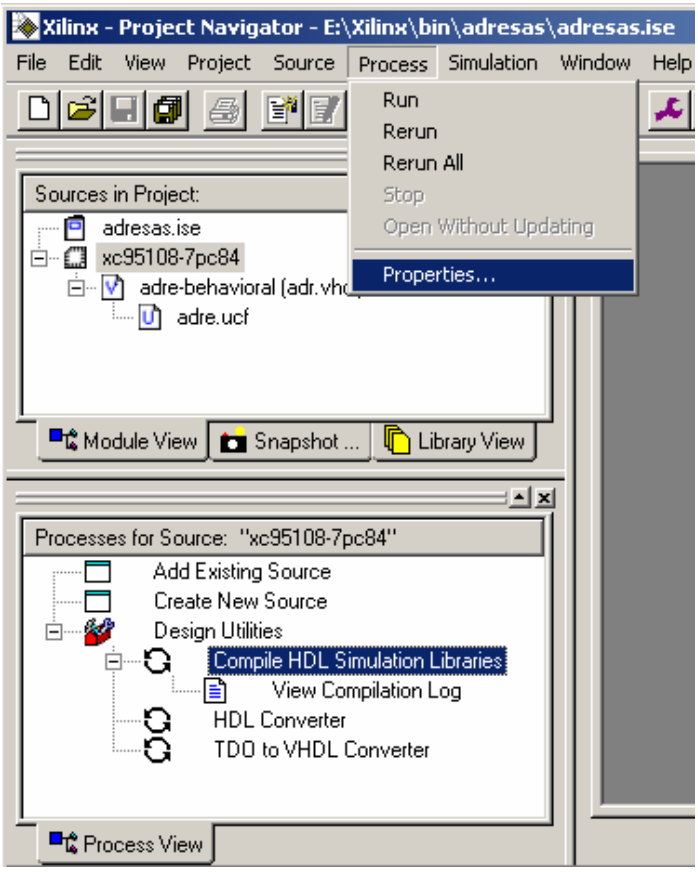


pav. 6-27

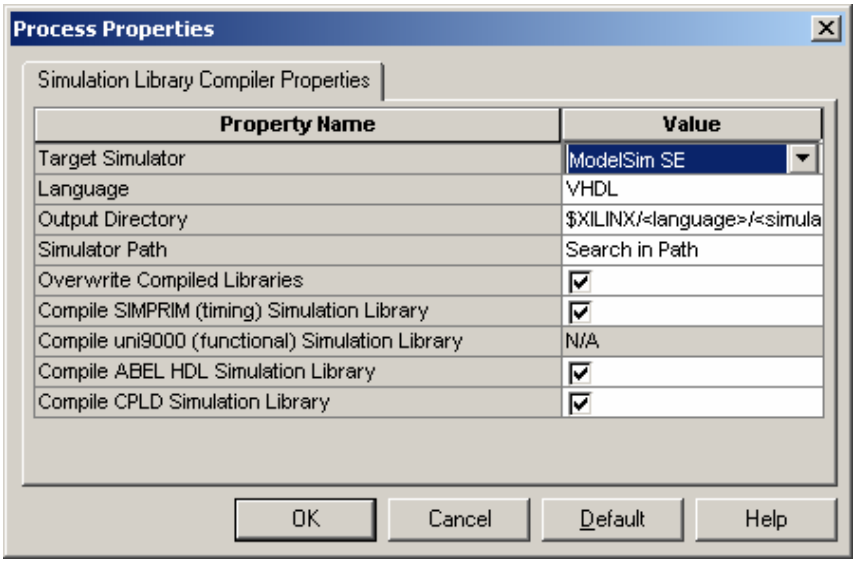
Taip paeiliui bus sukurtas VHDL kodo tuščias projektas su mikroschemos išvadų programinių aprašymu, toliau belieka tik gautą šabloną papildyti žemiau pateiktu VHDL kodu:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity adre is
    Port ( iej0,iej1,iej2,iej3,iej4,iej5,iej6,iej7: in std_logic;
           detect : out std_logic);
end adre;

architecture Behavioral of adre is

begin
process(iej0,iej1,iej2,iej3,iej4,iej5,iej6,iej7)

         begin
         if (iej0='0')and(iej1='1')and (iej2='0')and(iej3='1') and
(iej4='0')and(iej5='1') and (iej6='0')and(iej7='1') then
                  detect <= '1';
         else     detect <= '0';
         end if;

         end process;
end Behavioral;
```

Toliau pasirinkite (Process |Properties...) ir išsirinkite ModelSim SE|Target Simulator pav 6-28;29.



pav. 6-28



pav. 6-29

Tolesnis mikroschemos užprogramavimo procesas atliekamas taip kaip ir principinės schemos metodo atveju.

IŠVADOS

1. Projektuose naudojant JTAG sąsają turinčius programuojamus loginius prietaisus galima sutaupyti laiko, ploto ir pagerinti jų kokybę. Bet įrenginių savikaina yra didelė ir Lietuvos rinkai kol kas yra brangoka. Tačiau spartėjanti technologijų vystymąsis ilgainiui sumažina sąlyginai vieno loginio programuojamo ventilio kainą.
2. JTAG pagalba galima nekeičiant įrenginio aparatūrinės dalies atnaujinti sistemą pakeitus tik veikimo programą.
3. Naujausi JTAG įrenginiai palaiko perprogramavimą per LAN ir (arba) WAN tinklą nuotoliniu būdu. Tai reiškia, kad aparatūros gamintojas gali atnaujinti programinę įrangą pas vartotoją t.y. keisti aparatūros veikimo algoritmą., ištaisyti savo klaidas.
4. Lengvai galima keisti to paties lusto darbinę programą, neišmontuojant lusto iš montažinės plokštės.
5. Boundary scan įrenginiai yra naudingi sistemos projektavimui, derinimui ir gamykliniam testavimui.

LITERATŪRA

1. Alfred L. (1999)Crouch Design-for-Test for Digital IC's and Embedded Core Systems. Prentice Hall PTR..
2. Altera Corporation 1997 "Jam Programming & Test Language Specification" URL: http://www.altera.com/
3. ASSET InterTech, Inc. Texas Instruments Inc. URL: http://support.asset-intertech.com/
4. Balancing your design cycle. A practical guide to hw/sw co-verification. – URL: http://www.synopsys.com/.
5. Bob Roth(1992), "No 'Accounting' for BST," EDN Products & Careers Edition, Cahners Publishing Company, 275 Washington Street, Newton, MA 02158-1630, December , p. 65.
6. Bruce Peterson, Harry Jin(1993), "Testing High Performance Personal Computer Modules Using Boundary Scan: A Case Study," NEPCON West '93 Proceedings of the Technical Program, Reed Exhibition Companies, PO Box 5060, Des Plaines, IL 60017-5060, February, pp. 173-178.
7. Colin M. Maunder(1990), "Integrating Internal Scan Paths," The Test Access Port and Boundary'-Scan Architecture, IEEE Computer Society Press, 10662 Los Vaqueros Circle, PO Box 3014, Los Alamitos, CA 90720-1264.
8. Colin M. Maunder, Rodham E. Tulloss, ed.(1990), The Test Access Port and Boundary'-Scan Architecture, IEEE Computer Society Press, 10662 Los Vaqueros Circle, PO Box 3014, Los Alamitos, CA 90720-1264.
9. Craig A. Haller, Macraigor System Inc. "The ZEN of BDM" URL: http://www.macraigor.com/
10. Dan Romanchik(1993)"Getting Started With Boundary-Scan," Test & Measurement World, Cahners Publishing Company, 275 Washington Street, Newton, MA 02158-1630, March, pp. 75-78.
11. Dr. Bulent I. Dervisoglu, Mike Ricchetti(1992) "Design for Testability...Addressing the Problems Before They Occur," 1992 High Speed Digital Symposium, Hewlett-Packard, 4 Choke Cherry Road, Rockville, MD 20850, pp. 9-1 to 9-21.
12. ELECTRONIC INDUSTRIES ALLIANCE JEDEC Solid State Technology Association "Standard Test and Programming Language (STAPL)" URL: http://www.jedec.org/
13. Gerald Jacob(1994) "Let New Tools Add the Testability You Always Wanted," Evaluation Engineering, Nelson Publishing, 2504 N. Tamiami Trail, Nokomis, FL 34275, March pp. 56-59.

14. Gerald Jacob (1994) “Supporting the Boundless Growth of Boundary Scan,” Evaluation Engineering, Nelson Publishing, 2504 N. Tamiami Trail, Nokomis, FL 34275, July , pp. 58-62.
15. Harry Bleeker et al. (1993) Boundary-Scan Test: A Practical Approach. Kluwer Academic Publishers.
16. Harry Bleeker, Peter van den Eijnden, Frans de Jong (1994) Boundary-Scan Test - A Practical Approach, Kluwer Academic Publishers, 101 Philip Drive, Assinippi Park, Norwell, MA 02061.
17. Harry Jin, “Using Boundary-Scan Tests to Find Structural Faults at the Board Level,” Design & Test Expo 1993 Proceedings, Miller Freeman, 13760 Noel Road, Suite 500, Dallas, TX 75024, January 1993, pp. 321-327.
18. Hewlett-Packard, Application Note 1210-7, “Design for Testability Using Boundary-Scan 1149.1,” Hewlett-Packard, 4 Choke Cherry Road, Rockville, MD 20850, 1991.
19. IEEE Std 1149.1-1990 (includes IEEE Std 1149.1a-1993), IEEE Standard Test Access Port and Boundary-Scan Architecture, Institute of Electrical and Electronics Engineers, 345 East 47th Street, New York, NY 10017, October 1993.
20. J. Hirzer, “Testing Mixed Analog/Digital Ics(1990)” The Test Access Port and Boundary‘-Scan Architecture, IEEE Computer Society Press, 10662 Los Vaqueros Circle, PO Box 3014, Los Alamitos, CA 90720-1264.
21. Johnny M. Young (1994)“JTAG/IEEE 1149.1 Design Considerations,” Testability Products Data Book, Texas Instruments, PO Box 655303, Dallas, TX, 75265.
22. Jon Turino (1992)“You Can Obtain Boundary Scan’s Benefits Despite Use of Some Nonscan ICs,” EDN Magazine, Cahners Publishing Company, 275 Washington Street, Newton, MA 02158-1630, November 12, pp. 171-174.
23. Jusas.V.,Bareika. E., Šeinauskas R. (1997). Skaitmeninių sistemų projektavimas VHDL kalba. Kaunas: Technologija.
24. Kenneth P. Parker (1998) The Boundary-Scan Handbook, 2nd edition: Analog and Digital. Kluwer Academic Publisher.
25. Kenneth P. Parker (1992) The Boundary-Scan Handbook, Kluwer Academic Publishers, 101 Philip Drive, Assinippi Park, Norwell, MA 0206.
26. Lašas, Bartkevičius, Šurna “Pramoninė elektronika” V.
27. Mike Donlin,(1994) “Testing Dilemmas and Corporate Alliances Fuel Boundary Scan’s Acceptance,” Computer Design, PennWell Publishing Company, Ten Tara Blvd, Nashua, NH 03062-2801, January pp. 65-70.

28. Peter Fleming (1990) “Applications of IEEE Std 1149.1: An Overview,” The Test Access Port and Boundary‘-Scan Architecture, IEEE Computer Society Press, 10662 Los Vaqueros Circle, PO Box 3014, Los Alamitos, CA 90720-1264.
29. Peter Hansen (1993) “Boundary Scan Will Also Improve the Design Process,” Electronic Design, Penton Publishing, 1100 Superior Ave., Cleveland, OH 44114-2543, January 7, p. 109.
30. Raimundas Kirvaitis (1999) “Loginės schemos”: Enciklopedija Vilnius
31. Robert Dougherty (1993) “Applying IEEE 1149.1 Boundary Scan to the HAIDE System,” Design & Test Expo 1993 Proceedings, Miller Freeman, 13760 Noel Road, Suite 500, Dallas, TX 75024, January pp. 313-320.
32. Stan Runyon (1993) “Design-for-Test Finally Comes A-Board,” Electronic Engineering Times, CMP Publications, 600 Community Drive, Manhasset, NY 11030, January 11 pp. 39-46.
33. Stephen Yurash (1993) “A Practical Design Methodology for Running an Internal Full-Scan Test From the JTAG Port,” Design & Test Expo 1993 Proceedings, Miller Freeman, 13760 Noel Road, Suite 500, Dallas, TX 75024, January pp. 297-302.
34. Teradyne,(1992) VICTORY-BR05850A-0492, “VICTORY Boundary-Scan Test Software,” Teradyn, 321 Harrison Avenue, Boston MA 02118.
35. Texas Instruments, Literature Number SATT115A (1992) “ASSET Diagnostic System Product Family- Second Generation: Technical Overview and Applications,” Texas Instruments, PO Box 655303, Dallas, TX, 75265.
36. Texas Instruments, SN54/74ABT8996 Addressable Scan Port data sheet, Testability Products Data Book, Texas Instruments, PO Box 655303, Dallas, TX, 75265, 1994.
37. The Programmable Logic Data Book (1999). — Xilinx Inc.
38. The Programmable Logic Data Book (2000). — Xilinx Inc.
39. Wayne T. Daniel (1992) “Design Verification of a High Density Computer Using IEEE 1149.1,” International Test Conference 1992 Proceedings, International Test Conference, 514 E. Pleasant Valley Blvd., Suite 3, Altoona, PA 16602, September, pp. 84-90.
40. Xilinx XC9500 In-System Programmable CPLD Family Datasheet, v5.0 (09/99) http://www.xilinx.com
41. Гончаров Ю. Технология разработки eXpressDSP // Chip News. 2001. № 2. С. 26–31.
42. Использование интерфейса JTAG для отладки встраиваемых систем / Ключев А.О., Коровьякова Т.А., Платунов А.Е. // Изв. вузов. Приборостроение. 1998. Т41, №5 С. 45-50.

43. Кузелин М.(2001) ПЛИС фирмы Xilinx: семейство Spartan-II. — Компоненты и технологии, № 3.
44. Макаренко В. (2000) Методы внутрисхемного тестирования в производстве электронной техники // Электронные компоненты и системы. № 10.
45. Мальцев. П.П., Гарбузов. Н.И., Шарапов. А.П., Кнышев. (1998) Д.А. Программируемые логические ИМС на КМОП-структурах и их применение. — М.: Энергоатомиздат.
46. Платунов А., Постников Н., Чистяков А. Механизм граничного сканирования в неоднородных микропроцессорных системах // Chip News. 2000. № 10. С. 8–13.
47. Программируемые логические интегральные схемы фирмы Xilinx.(2000) Серия FAST FLASH CPLD. Краткое техническое описание семейств XC9500 и XC9500XL. — М.: ЗАО «SCAN».
48. Программируемые логические интегральные схемы фирмы Xilinx. (2000)Серия VIRTEX. Краткое техническое описание. — М.: ЗАО «SCAN».
49. Угрюмов Е.П. Цифровая схемотехника. СПб.: БХВ – Санкт–Петербург, 2000. – 528 с.: ил. ARM Limited 1999 “Using Embedded ICE” URL: http://www.arm.com/
50. ftp://ftp.xilinx.com/pub/ftpfiles.htm
51. http://aics.ru/wbooks/2/280.htm
52. http://aics.ru/wbooks/2/286.htm
53. http://cn.chipinfo.ru/archive/chipnews/200106/4.html
54. http://lmt.cs.ifmo.ru/publications.html
55. http://plisdeveloper.narod.ru/WebPack/WEBPACK.htm
56. http://toolbox.xilinx.com/docsan/xilinx4/data/docs/pac/appendixa2.html
57. http://tutor.al-williams.com
58. http://www.actel.com/
59. http://www.altera.com/
60. http://www.chipnews.com.ua/html.cgi/arhiv/99_08/stat_2.htm
61. http://www.orc.ru/~dkuzn/j_intro.htm
62. http://www.telesys.ru/wwwboards/fpga
63. http://www.xilinx.com