



VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS

ELEKTRONIKOS FAKULTETAS

KOMPIUTERIŲ INŽINERIJOS KATEDRA

Vaiva Šlegerytė

**PROGRAMUOJAMOS LOGIKOS LUSTO TAIKYMAS KOMPIUTERYJE
NAUDOJANT USB SĄSAJĄ**
PROGRAMMABLE LOGIC DEVICE APPLICATION IN PC USING USB INTERFACE

Baigiamasis magistro darbas

Elektronikos inžinerijos studijų programa, valstybinis kodas 621H61003

Mikro ir nano elektronikos specializacija

Elektronikos inžinerijos studijų kryptis

Vilnius, 2013

VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS

ELEKTRONIKOS FAKULTETAS

KOMPIUTERIŲ INŽINERIJOS KATEDRA

TVIRTINU
Katedros vedėjas

(parašas)

Algirdas Baškys

(Vardas, pavardė)

(Data)

Vaiva Šlegerytė

PROGRAMUOJAMOS LOGIKOS LUSTO TAIKYMAS KOMPIUTERYJE

NAUDOJANT USB SĄSAJĄ

PROGRAMMABLE LOGIC DEVICE APPLICATION IN PC USING USB INTERFACE

Baigiamasis magistro darbas

Elektronikos inžinerijos studijų programa, valstybinis kodas 621H61003

Mikro ir nano elektronikos specializacija

Elektronikos inžinerijos studijų kryptis

Vadovas doc. dr. Giedrius Masalskis

(Moksl. laipsnis, vardas, pavardė)

(Parašas)

(Data)

Konsultantas lekt. Vaida Buivydienė

(Moksl. laipsnis, vardas, pavardė)

(Parašas)

(Data)

Vilnius, 2013

Vilniaus Gedimino technikos universitetas
Elektronikos fakultetas
Kompiuterių inžinerijos katedra

ISBN ISSN
Egz. sk.
Data-.....-.....

Antrosios pakopos studijų **Elektronikos inžinerijos** programos baigiamasis darbas 4

Pavadinimas **Programuojamos logikos lusto taikymas kompiuteryje naudojant USB sąsają**

Autorius **Vaiva Šlegerytė**

Vadovas **dr. Giedrius Masalskis**

Kalba: lietuvių

Anotacija

Baigiamajame magistro darbe nagrinėjamas programuojamos logikos lusto panaudojimas kompiuteryje per USB sąsają. Tiriamos programuojamo lusto galimybės siekiant panaudoti lusto funkcijas kompiuterio programinėje įrangoje. Išnagrinėti programuojamos logikos įrenginiai, USB-UART sąsaja. Aptarta galimybė panaudoti didelės sparto (iki 3000000 bodų/s) nuoseklią sąsają duomenims tarp kompiuterio ir programuojamos logikos lusto siųsti. Suprojektavus USB-UART modulį atliktas duomenų perdavimo tyrimas. Nustatyta, kad galima pasiekti maksimalią duomenų perdavimo spartą, tačiau norint gauti nesugadintus duomenis reikia naudoti programinį klaidų taisymo protokolą. Darbe pateikiama rezultatų analizė, pateikiami pasiūlymai tolimesniems tobulinimams.

Darbą sudaro 3 dalys: įvadas, analitinė dalis, taikomoji dalis, tiriamoji dalis, išvados ir siūlymai, literatūros sąrašas.

Darbo apimtis – 32 p. teksto be priedų, 21 iliustr., 6 lent., 25 bibliografiniai šaltiniai.

Atskirai pridedami darbo priedai.

Prasminiai žodžiai: programuojamos logikos lustas, VHDL, USB-UART sąsaja, duomenų perdavimas

Vilnius Gediminas Technical University
Faculty of Electronics
Department of Computer Engineering

ISBN ISSN
Copies No.
Date*.....*

Master Degree Studies **Electronics Engineering** study programme Final Work 4

Title **Programmable Logic Device Application in PC Using USB Interface**

Author **Vaiva Šlegerytė**

Academic supervisor **Dr Giedrius Masalskis**

Thesis language:
Lithuanian

Annotation

Possibilities of programmable logic device applications in PC using USB interface were investigated in master thesis. FPGA devices and USB-UART interface was analyzed. UART interface was created in FPGA using VHDL in order to test the possibilities of high speed UART interface. Analysis of data transfer was made to find out the fastest baud rate with low bit error rate. It was estimated that it is possible to transfer data on 1500000 bauds/s with low bit error rate. This allows sending data on 1.2 Mb/s speed. In order to get 3000000 bauds/s data rate, data error detection protocol must be created on computer's side.

Structure: introduction, analytical part, implementation part, research part, conclusions and suggestions, references.

Thesis consists of: 32 p. text without extras, 21 pictures, 6 tables, 25 bibliographical entries.

Appendixes included.

Keywords: FPGA, VHDL, USB-UART interface, data transfer

(Baigiamojo darbo sąžiningumo deklaracijos forma)

VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS

Vaiva Šlegerytė, 20076111

(Studento vardas ir pavardė, studento pažymėjimo Nr.)

Elektronikos fakultetas

(Fakultetas)

Elektronikos inžinerija, MNEfm-11

(Studijų programa, akademinė grupė)

**BAIGIAMOJO DARBO (PROJEKTO)
SĄŽININGUMO DEKLARACIJA**

2013 m. birželio 6 d.

Patvirtinu, kad mano baigiamasis darbas tema „Programuojamos logikos lusto taikymas kompiuteryje naudojant USB sąsają“ patvirtintas 2011 m. lapkričio 22 d. dekanų potvarkiu Nr. 260el, yra savarankiškai parašytas. Šiame darbe pateikta medžiaga nėra plagijuota. Tiesiogiai ar netiesiogiai panaudotos kitų šaltinių citatos pažymėtos literatūros nuorodose.

Parėnkant ir įvertinant medžiagą bei rengiant baigiamąjį darbą, mane konsultavo mokslininkai ir specialistai: Vaida Buivydienė. Mano darbo vadovas dr. Giedrius Masalskis.

Kitų asmenų indėlio į parengtą baigiamąjį darbą nėra. Jokių įstatymų nenumatytų piniginių sumų už šį darbą niekam nesu mokėjęs (-usi).

(Parašas)

Vaiva Šlegerytė

(Vardas ir pavardė)

TURINYS

SANTRUMPOS	8
ĮVADAS	9
1. LPLM ĮRENGINIŲ IR USB SĄSAJOS ANALITINĖ APŽVALGA	10
1.1. Lauku programuojamos loginės matricos	10
1.2. USB-UART sąsajos analizė	18
2. USB SĄSAJOS MODULIO KŪRIMAS	22
2.1. Naudojama aparatinė ir programinė įranga	22
2.2. USB-UART modulio LPLM luste kūrimas	27
2.3. Komunikacijos programos kompiuterio OS kūrimas	34
3. DUOMENŲ PERDAVIMO TARP LPLM IR KOMPIUTERIO TYRIMAS	36
3.1. Ryšio kokybės nustatymas	36
3.2. USB-UART modulių palyginimas	39
3.3. Pasiūlymai tobulinimui	40
APIBENDRINIMAS IR IŠVADOS	41
ŠALTINIAI IR LITERATŪRA	42
PRIEDAI	44
A Priedas.	45
B priedas.	48
C priedas.	50
D priedas.	52

SANTRUMPOS

- DCM Skaitmeninis laikmačio valdiklis (angl. *Digital Clock Manager*)
- IN Intelektinės nuosavybės blokas (angl. *Intellectual Property Core*)
- KLB Konfigūruojamas loginis blokas (angl. *Configurable Logic Block*)
- LPLM Lauku programuojamos loginė matrica (angl. *Field Programmable Gate Array*)
- PL Paieškos lentelė (angl. *Look-Up-Table*)
- PLL Fazės derinimo kilpa (angl. *Phase-locked loop*)
- SPIG Specialios paskirties integrinė grandinė (angl. *Application-Specific Integrated Circuit*)

IVADAS

LPLM lustuose realizuotos grandinės yra didelės veikimo spartos ir lankstumo, dėl to programuojamos logikos testavimo plokštės dažnai naudojamos prototipams gaminti. Taip pat LPLM taikomos sistemose, kuriose reikalingas nuolatinis atnaujinimas (angl. *update*). Bene svarbiausias modulis bet kuriam periferiniam įrenginiui yra duomenų perdavimo sąsaja. Nuo jos priklauso periferinio įrenginio ryšio su kompiuteriu patikimumas. Šiuo metu bet kuriai elektroninei sistemai ryšys su kompiuteriu yra kritiškai svarbus, dėl to reikia siekti sukurti kokybišką, spartų, tikslų ir optimalų duomenų perdavimo modulį. Galimybė nesudėtingai susieti asmeninį kompiuterį su LPLM įrenginiu leistų gaminti galingus ir sparčius prietaisus, skirtus specializuotoms operacijoms, tokioms kaip duomenų šifravimas, vykdyti.

Ryšio kanalui tarp kompiuterio ir LPLM lusto sudaryti gali būti naudojamas USB, USB 2.0, USB 3.0 arba USB-UART protokolas. USB prievadas pasirinktas todėl, kad yra plačiai paplitęs. USB-UART sąsaja yra nesudėtingai realizuojama, ir tinkama naudoti nedideliems duomenų srautams perduoti sistemai, esančiai LPLM luste. USB-UART modulio realizacija yra nesudėtinga, lyginant su USB protokolu, be to toks modulis užima nedaug LPLM resursų, todėl luste galima realizuoti sudėtingus duomenų apdorojimo algoritmus. Įprasta, kad USB-UART sąsaja duomenys perduodami nedideliu greičiu: 115200 bodų/s, 57600 bodų/s ar mažesniu. Aparatinė įranga leidžia pasiekti daug didesnę duomenų perdavimo spartą – iki 3000000 bodų/s. Tokios duomenų perdavimo spartos visiškai pakanka didesniems duomenų srautams į LPLM lustą perduoti. Keliamas darbo tikslas yra USB sąsajos įdiegimo programuojamos logikos luste galimybių tyrimas, siekiant panaudoti lusto funkcijas kompiuterio programinėje įrangoje. Siekiant įgyvendinti darbo tikslą vykdomi uždaviniai: USB sąsajos ir programuojamos logikos įrenginių analizė, USB sąsajos palaikymo modulio kūrimas, programinės įrangos kūrimas bei sąsajos tyrimas.

USB-UART sąsaja yra labai plačiai taikoma, tačiau pačios sąsajos savybės yra mažai tyrinėtos. LPLM suteikia galimybę realizuoti skirtingos architektūros USB-UART modulius ir juos plačiau ištirti. Tyrimų rezultatai ir išvados sudaro prielaidas greitam, nesudėtingam ir efektyviam USB-UART moduliui pasiekti. Atlikta analizė leidžia teikti pasiūlymus tolimesniems sąsajos tyrimams ir patobulinimams. Tai leistų pašalinti problemas, kylančias kuomet reikia naudoti paprastą bet efektyvų ir patikimą duomenų perdavimo modulį.

1. LPLM ĮRENGINIŲ IR USB SĄSAJOS ANALITINĖ APŽVALGA

Kompiuterijoje ir elektronikoje skaičiavimams atlikti naudojama arba programinė arba aparatinė įranga. Aparatinę įrangą paprastai sudaro specialios paskirties integrinės grandinės (SPIG). SPIG lustai yra optimizuoti specialiems skaičiavimams atlikti. SPIG lustai gali būti skirti skaitmeniniams signalų apdorojimui, kodavimo ir dekodavimo operacijoms atlikti, vaizdui apdoroti ir pan. Tokie lustai maksimaliai greitai ir kokybiškai atlieka jiems būdingus skaičiavimus, tačiau pagrindinis trūkumas yra jų orientacija tik į vieno tipo operacijas. SPIG lusto projektavimo ir gamybos kaina siekia milijonus JAV dolerių. Programinė įranga priešingai – pasižymi dideliu lankstumu. Naudojant programinę įrangą galima spręsti daug skirtingų uždavinių, programinę įrangą nesudėtinga modifikuoti. Pagrindinis programinės įrangos trūkumas – skaičiavimai neatliekami taip sparčiai kaip aparatinėje įrangoje. Taip pat programinė įranga paprastai reikalauja daug galios resursų [1].

1985 metais rinkoje pasirodžius pirmiesiems komerciniams lauku programuojamųjų loginių matricų (LPLM) produktams, pagamintiems kompanijos *Xilinx* požiūris į skaitmeninės elektronikos projektavimą bei kūrimą pasikeitė iš esmės [2]. XC2064 buvo pirmasis LPLM lustas, kuriame pritaikyti visiškai nauji ir radikalūs tam laikotarpiui sprendimai. Atsirado galimybės viename luste suderinti aparatinės ir programinės įrangos privalumus. LPLM lustai leidžia įgyvendinti skaitmenines grandines, kurios pasižymi visomis aparatinei įrangai būdingomis savybėmis – mažos energijos sąnaudos, didelė veikimo sparta, sąlyginai kompaktiškas lusto užimamas plotas. Be to LPLM lustai gali būti perprogramuojami, t. y. galima keisti jų konfigūraciją. Tai leidžia įgyvendinti skirtingus sprendimus, esant reikalui juos atnaujinti ar kardinaliai pakeisti. Be to, tokios sistemos gali veikti žymiai sparčiau nei mikrovaldikliai [3].

Akivaizdu, kad LPLM lustuose realizuotos skaitmeninės sistemos yra labai lanksčios. Įvairios programinės priemonės leidžia paspartinti sistemos projektavimo procesą. Tam gali būti naudojami programiniai procesoriai, pavyzdžiui *MicroBlaze*, arba intelektinės nuosavybės (IN) blokai, komponentų aprašai. Šios priemonės sutrumpina laiką, skirtą sudėtingos sistemos projektavimui ir leidžia nesudėtingai kurti ir testuoti specializuotus perprogramuojamus lustus, pavyzdžiui, skirtus naudoti kriptografijai [4].

1.1. Lauku programuojamos loginės matricos

LPLM yra lustai, kurios galima užprogramuoti jau po pagaminimo. Tai leidžia greitai ir efektyviai taisyti sistemos klaidas ar ją atnaujinti. Dėl didelės veikimo spartos LPLM puikiai tinka dideliems duomenų srautams, kuriems reikalingi lygiagretūs procesai, apdoroti. LPLM puikiai

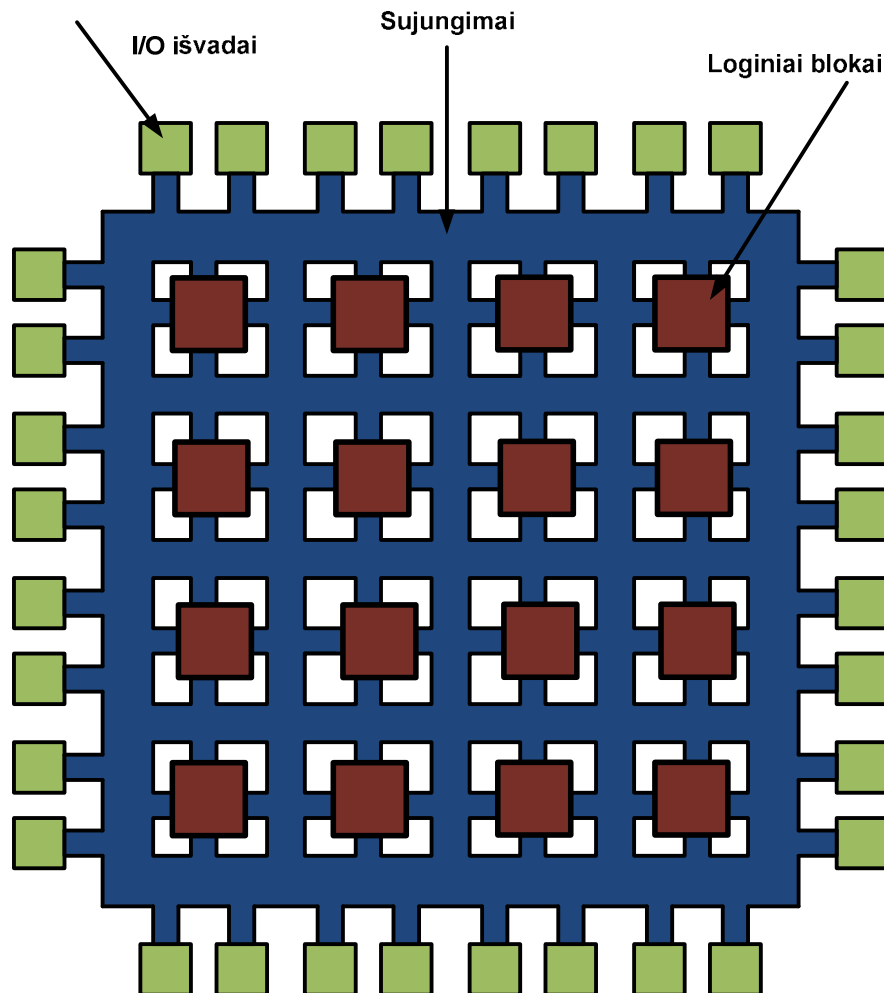
susidoroja su tokiomis užduotimis kaip neuroniniai tinklai, vaizdo apdorojimas ar kriptografinės operacijos [5]. Pagrindinis LPLM trūkumas – didelis lusto plotas, dėl to LPLM lusto įsigijimo kaina stipriai viršija jau pagaminto SPIG lusto įsigijimo kainą. Nepaisant to, projektuojant specializuotas sistemas, skirtas smulkiaserijinei gamybai, šie trūkumai tampa ne esminiai.

LPLM yra programuojamos logikos ir tarpusavio ryšių struktūros rinkinys. Dėl išskirtinės LPLM architektūros galima greitai sumodeliuoti ir įdiegti norimą loginį grandyną į LPLM lustą. Tam naudojama speciali programinė įranga. Šios priemonės ir LPLM išskirtinės savybės leidžia išvengti didelių kaštų, kurie būdingi projektuojant SPIG sistemas. LPLM luste įdiegtoje sistemoje galima labai greitai ištaisyti projekto klaidas – pakanka modifikuoti lusto konfigūraciją. Tai leidžia LPLM naudoti kuriant loginių schemų prototipus. Be to, LPLM gali būti puikiu standartinės logikos ar loginių masyvų pakaitalu. Toks LPLM universalumas ir lankstumas turi trūkumų. Dėl specifinės architektūros LPLM lustai užima didelį plotą, dėl ko išauga lusto kaina, padidėja suvartojamoji galia. Dėl didelio ploto didėja vėlinimai grandinėse, dėl to norint didinti veikimo spartą, tenka kelti taktinį dažnį. Dažnio didinimas lemia dar labiau padidėjusias energijos sąnaudas.

Siekiant spręsti išskylančias problemas įsisavinamos naujos technologijos. Pavyzdžiui, šiuo metu kompanija *Xilinx* rinkai siūlo LPLM lustus, pagamintus 28 nm technologija, kurioje po metaliniu tranzistoriaus užtūros elektrodu formuojamas didelės dielektrinės skvarbos izoliacinis sluoksnis [6]. Taip pat naudojami nauji junginiai sujungimams formuoti. Technologija leidžia sumažinti pasyvias energijos sąnaudas iki 50 % lyginant su ankstesnėmis technologijomis. Be to, siekiant padidinti LPLM lustų galimybes taikomos įvairios heterogeninių struktūrų formavimo technologijos, kai sujungimai ir loginiai elementai formuojami atskirai ir vėliau jungiami į vieną kelių aukštų struktūrą. Tai sumažina signalo vėlavimą, pagerina našumą [7]. Perėjimas prie 28 nm technologijos leido sumažinti santykinę nuotėkio galią 50 %, lyginant su prieš tai naudota technologija [8]. 2012 metais kompanija *Xilinx* rinkai pasiūlė naujos kartos LPLM lustus, kuriuose pritaikytos naujausios technologijos: heterogeninės struktūros, 28 nm technologija, aukštų dažnių ryšio elementų integracija ir kt. Toks žingsnis rodo, kad FPGA svarba kuriant įterptines sistemas auga labai sparčiai. Technologijos orientuojamos į galios mažinimą ir heterogeninių sistemų viename luste ir kūrimą. Skirtingų elementų (loginių grandinių, aukštų dažnių siųstuvų ir imtuvų, optinių skaidulų ir kt.) integracija viename korpuse yra viena iš pagrindinių nano elektronikos vystymosi sričių. Heterogeniniai LPLM lustai suteikia galimybes išbandyti ir įvertinti naujausius elektronikos technologijų pasiekimus.

LPLM architektūros pagrindas – trys skirtingos grupės konfigūruojamų elementų: loginių blokų, įvesties/išvesties išvadų (blokų) ir sujungimų [1]. Supaprastinta LPLM lusto blokinė schema

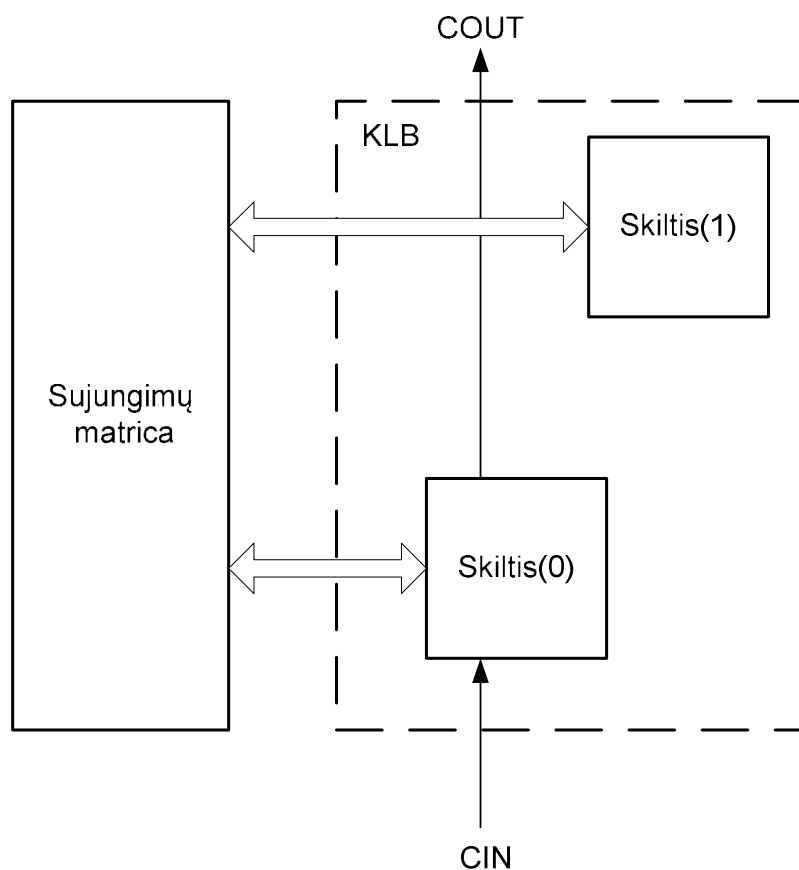
pavaizduota 1 paveiksle. Konfigūruojamus loginius blokus sudaro kombinacinės ir nuosekliosios loginės schemas ar elementai, skirti realizuoti loginius elementus ir registrus. Viena loginiame bloke gali būti realizuota bet kuri loginė funkcija, kurios įėjimų skaičius priklauso nuo LPLM architektūros. Bazinis sandaros modelis yra būdingas visiems LPLM lustams, tačiau vidinė konfigūruojamų blokų struktūra ar luste esančių loginių elementų skaičius priklauso nuo gamintojo ir lusto architektūros. Atliekant magistro darbą buvo naudojama plokštė su *Spartan-6* architektūros LPLM lustu, todėl šią architektūrą nagrinėsime plačiau.



1 pav. LPLM lusto sandaros schema [9]

Konfigūruojami loginiai blokai yra pagrindiniai elementai, leidžiantys realizuoti tiek kombinacines tiek nuoseklias logines schemas. Paprastai, loginiame bloke galima realizuoti 5–6 įėjimus turinčias Bulio funkcijas [1]. Įvesties/išvesties blokai tarpusavyje susieja lusto grandinę su išoriniais prievadais. Konfigūruojami sujungimai leidžia loginius blokus susieti tarpusavyje, taip

formuojant reikalingą struktūrą. *Spartan-6* architektūroje kiekvienas loginis blokas yra susietas su sujungimų matrica, kuri leidžia susieti konkretų KLB su kitais LPLM luste esančiais blokais. Struktūrinė KLB schema pavaizduota 2 paveiksle. Kiekvieną KLB sudaro 2 skiltys (angl. *slice*), kurios nesusungtos tarpusavyje.



2 pav. KLB blokinė schema [10]

Pagrindiniai skilčių elementai yra paieškos lentelės (PL) ir trigeriai, tačiau efektyviai dirbančiam loginiam blokui jų nepakanka. *Spartan-6* architektūroje kiekviena skiltis turi 4 paieškos lenteles kombininės logikos elementams realizuoti ir 8 trigerius informacijai saugoti. Bazinė skiltis SLICEX yra sudaryta tik iš PL ir trigerių. Skiltyse gali būti naudojami įvairūs logikos elementai, tokie kaip multiplekseriai, sumatoriai ar atminties elementai. SLICEL tipo skiltyse naudojami papildomi kombininės logikos elementai – daugiafunkciniai multiplekseriai bei aritmetinės pernašos struktūros. Tuo tarpu SLICEM tipo ląstelė papildoma 64 bitų paskirstytąjį RAM atmintine bei kintamo dydžio poslinkio registru. Maksimalus galimas registro dydis yra 32 bitai. Skilčių tipai ir jas sudarantys elementai plačiau aptarti 1 lentelėje. XC6SLX16 lustas daugiausiai turi SLICEX tipo skilčių. SLICEL ir SLICEM tipo skilčių yra dvigubai mažiau.

Paprastai išlaikomas toks skilčių santykis: 50% SLICEX, 25% SLICEL bei 25% SLICEM. SLICEX tipo skiltys sudarytos tik iš bazinių logikos elementų: PL, kurios aprašo kombinačines funkcijas ir trigerių, kurie veikia kaip atminties elementai [10]. SLICEL papildomai turi multipleksorius ir pernašos logiką, o SLICEM skiltis yra papildyta RAM atmintimi ir poslinkio registrais. Toks elementų paskirstymas leidžia efektyviai išnaudoti lusto galimybes.

1 lentelė. *Spartan-6* architektūros skiltyse naudojami elementai [10]

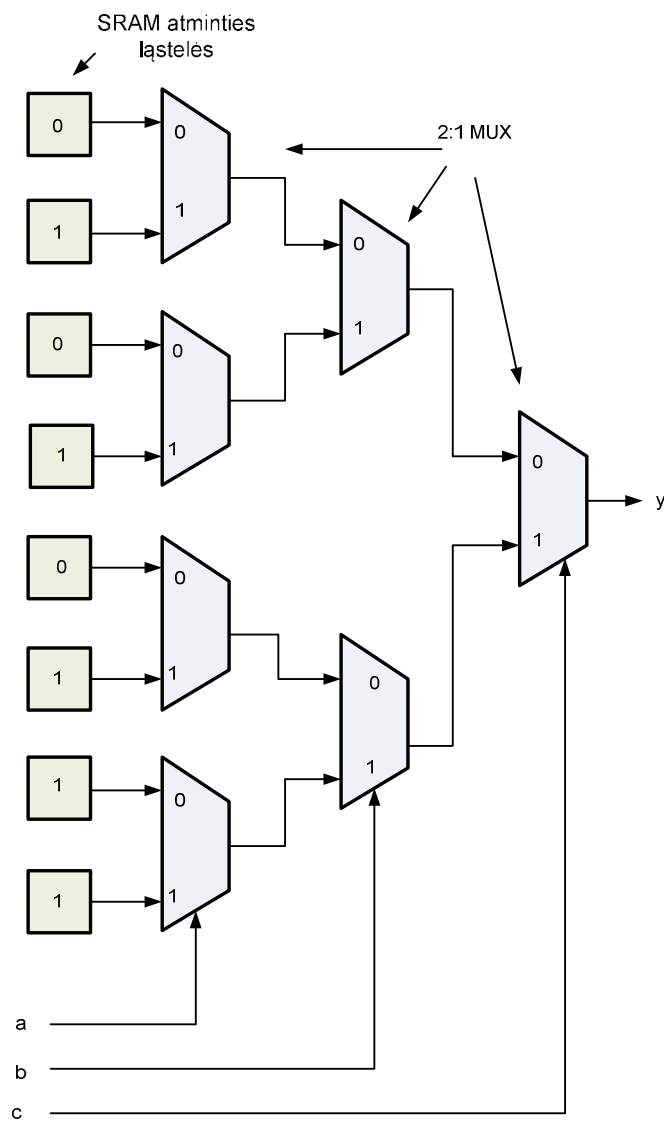
Elementas	SLICEX	SLICEL	SLICEM
6 įėjimų PL	yra	yra	yra
8 trigeriai	yra	yra	yra
Daugiafunkciniai multiplekseriai	nėra	yra	yra
Pernašos logika	nėra	yra	yra
Paskirstytoji laisvosios kreipties atmintis (angl. <i>RAM</i>)	nėra	nėra	yra
Poslinkio registrai	nėra	nėra	yra

2 lentelėje pateikiama *Spartan-6* šeimos LPLM lustų palyginimas. Nagrinėjant duomenis matyti, kad prieš tai aptartas skilčių santykis yra išlaikytas. *Nexys3* plokštėje naudojamas lustas XC6SL16. Šis lustas turi 2278 skiltis, iš kurių pusė yra SLICEX tipo. SLICEL tipo skilčių yra šiek tiek daugiau nei SLICEM, tačiau pagrindinės proporcijos yra išlaikytos. Visi *Spartan-6* šeimos programuojamos logikos lustai turi paminėtas trijų tipų skiltis. Vienintelė išimtis yra XC6SLX4 lustas. Jis turi tik SLICEM ir SLICEX tipų skiltis. Abiejų tipų skilčių luste yra po lygiai. Tai paprasčiausias *Spartan-6* šeimos lustas, turintis mažiausiai elementų. Tuo tarpu talpiausias lustas XC6SLX150 turi 10 kartų daugiau elementų, nei darbe naudojamas XC6SL16.

2 lentelė. XCS6LX16 elementų sąrašas [10]

Lustas	SLICEM skiltys	SLICEL skiltys	SLICEX skiltys	PL skaičius	Pask. RAM	Poslinkio registrai	Trigerių skaičius
XC6SLX4	300	0	300	2400	75	38	4800
XC6SLX16	544	595	1139	9112	136	68	18224
XC6SLX150	5420	6099	11519	92152	1355	678	184304

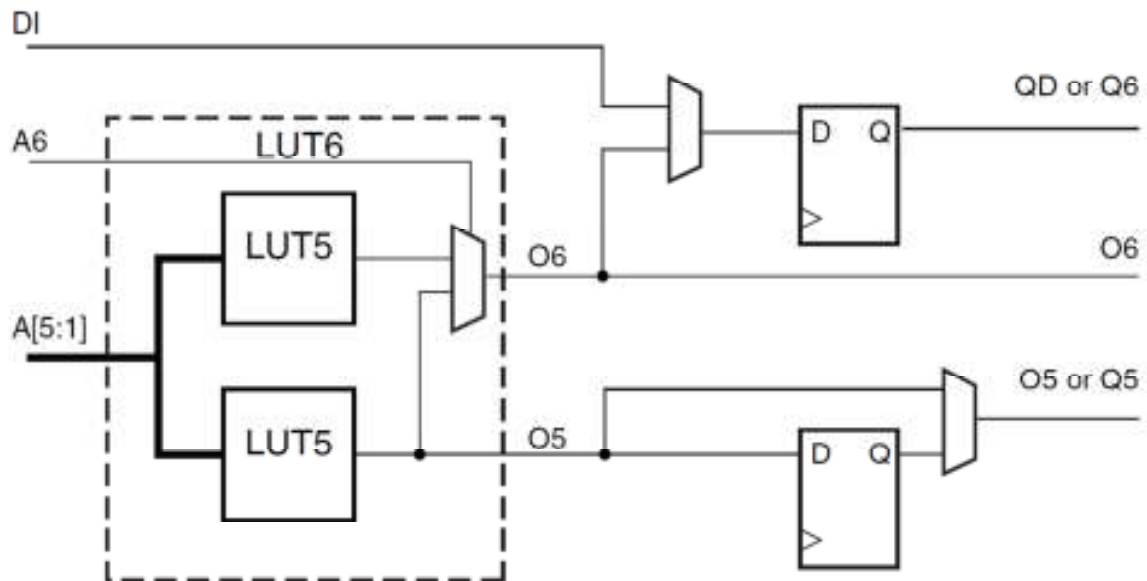
Baziniai visų skilčių elementai yra PL ir trigeriai. Bet kokius skaičiavimus galima išreikšti Bulio (loginėmis) funkcijomis, kurios gali būti aprašytos teisingumo lentele. Bulio funkcijos yra tarsi statybiniai blokai, iš kurių galima sudaryti sudėtingas struktūras, atliekančias matematines operacijas, pavyzdžiui, sumatorius, daugybos įrenginius, įvairius skaitiklius ir kt. Paieškos lentelė gali būti sukonfigūruota bet kokiai loginei funkcijai. Schemotechniškai paieškos lentelės realizuojamos sujungiant atminties elementus (naudojama SRAM tipo atmintis) ir multipleksorius. SRAM yra statinė laisvosios kreipties atmintis. Statine ji vadinama dėl to, kad ji yra stabili, t.y. informacijos nereikia atnaujinti. Informacija SRAM atmintyje išlieka tol, kol neatjungiamas maitinimas.



3 pav. PL veikimas [11]

PL struktūra ir veikimas pavaizduoti 3 paveiksle. LPLM lusto konfigūracijos metu į SRAM elementus įrašomos Bulio funkcijų teisingumo lentelės reikšmės. Multiplexeriai komutuoja funkcijos įėjimus (a, b, c) ir priklausomai nuo jų reikšmių išrenkamas reikiamas funkcijos rezultatas. Tokia struktūra yra labai efektyvi, tačiau pagrindinis jos trūkumas – užimamas plotas. Vienam SRAM tipo atminties elementui realizuoti reikia 6 tranzistorių. Multiplexeris yra dar sudėtingesnis elementas, susidedantis iš kombinacinės logikos elementų ARBA ir iverterių. Dėl to LPLM lusto kaina ženkliai išauga. Realizuojant tą pačią Bulio funkciją SPIG luste ji užimtų žymiai mažesnę kristalo plotą.

Kaip jau minėta, *Spartan-6* architektūroje naudojamos PL turinčios 6 įėjimus [10]. Paieškos lentelės schema *Spartan-6* lustuose, pavaizduota 4 paveiksle. PL turi 6 nepriklausomus įėjimus (A1 – A6) ir 2 nepriklausomus išėjimus (O5-O6) kiekvienam funkciniam generatoriui, esančiam skiltyje. Funkciniai generatoriai gali įgyvendinti bet kurią 6 įėjimų Bulio funkciją. Be to, kiekvienas generatorius taip pat gali įgyvendinti dvi 5 įėjimų Bulio funkcijas, jeigu jos turi bendrą įėjimo signalų. Kai funkcinis generatorius naudojamas 6 įėjimų Bulio funkcijai realizuoti, naudojamas tik O6 išėjimas, tuo tarpu sudarant 5 įėjimų funkcijas naudojami abu išėjimo signalai. Šiuo atveju išvade A6 automatiškai nustatomas aukštas signalo lygis. PL nepriklausomi išėjimai naudojant multiplexerius sujungti su trigeriais, kurie naudojami, kai reikia sudaryti nuoseklias logines schemas.

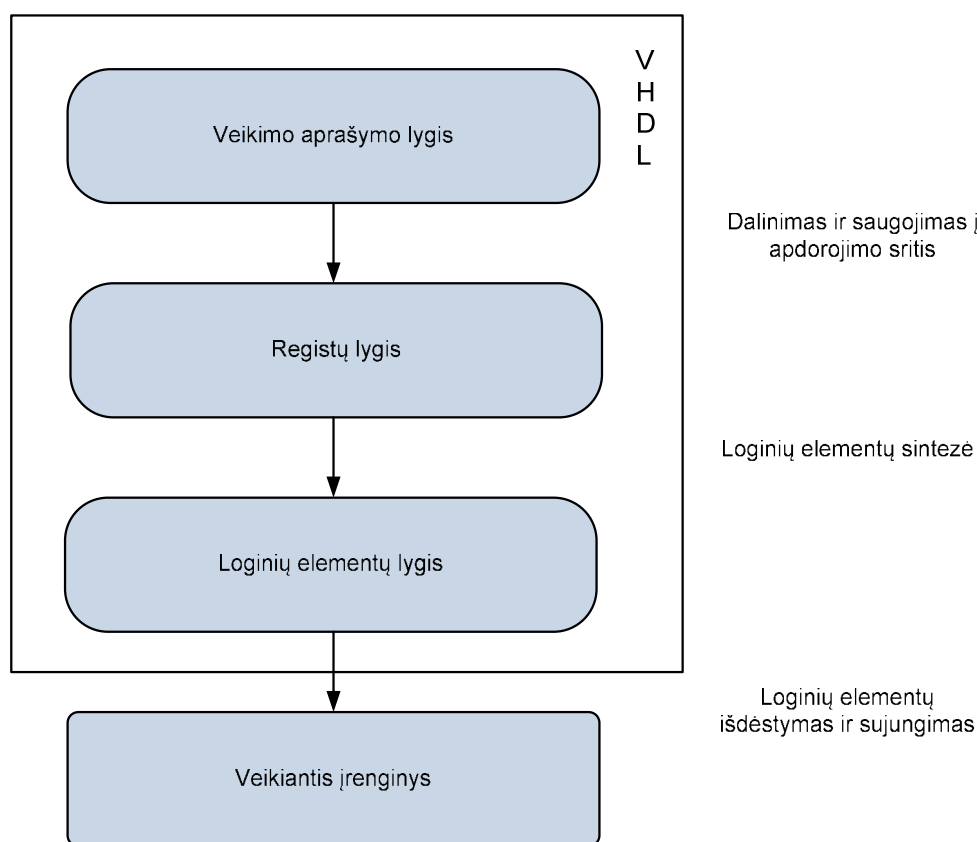


4 Pav. *Spartan-6* architektūros LPLM lusto PL [10]

Konfigūruojami sujungimai skirti ryšiams tarp loginių blokų, įvesties/išvesties blokų, atminties ląstelių ir kitų komponentų sudaryti [1]. Sujungimų matrica realizuojama naudojant

laidininkus ir programuojamus raktus. Rakto įėjimo reikšmė įrašoma į atminties elementą. Siekiant sumažinti signalo vėlinimą dėl didelio lusto ploto naudojami kelių tipų sujungimai jungiantys loginių blokų įėjimus su išėjimais, gretimus loginius blokus tarpusavyje, jungiantys kas antrą loginį bloką arba kas ketvirtą loginį bloką. Įvairių tipų magistralės naudojamos siekiant optimizuoti loginės grandinės struktūrą bei sumažinti vėlinimo laikus.

LPLM yra programuojamos naudojant specialiai tam skirtas programavimo kalbas, tokias kaip *Verilog* ar *VHDL*. Naudojant šias kalbas galima nesudėtingai realizuoti įvairaus pobūdžio logines schemas. *VHDL* šiuo metu laikoma populiariausia programavimo kalba, skirta integrinių grandynų projektavimui. Ši kalba yra standartizuota ir nuolat tobulinama. Integrinio grandyno įgyvendinimo *VHDL* programavimo kalba etapai parvaizduoti 5 paveiksle.



5 pav. Integrinio grandyno įgyvendinimo VHDL kalba etapai [12]

Visų pirma yra aprašomas schemos veikimas. Aprašoma gali būti tiek schema, tiek algoritmai, tiek vykdomos funkcijos. Toliau specializuoti įrankiai kiekvieną kodo dalį keičia į registrų lygį, kur yra dalinama į saugojimo ir apdorojimo sritis. Žemiausias lygis – loginių elementų

lygis. Šiame lygmenyje elementai išdėstomi pagal LPLM žemėlapi ir iš esmės yra tinkami LPLM lusto konfigūracijai. Neretai iškyla problemų, kai sintezės įrankis nesugebą funkcinio aprašymo paversti į loginių elementų seką. Tuomet reikia ieškoti specifinių sprendimų, dažnai priklausančių nuo LPLM lusto architektūros.

VHDL programavimo kalba galima naudoti tiek nesudėtingoms loginėms grandinėms aprašyti, tiek sudėtingoms schemoms sudaryti. Be to, *Xilinx*. siūlomos programinės priemonės (*ISE Design Suite*) leidžia ne tik modeliuoti schemas naudojantis šiomis programavimo kalbomis, bet ir sudaryti norimas struktūras jungiant loginius elementus tarpusavyje iš jų gaunant norimą programinį kodą.

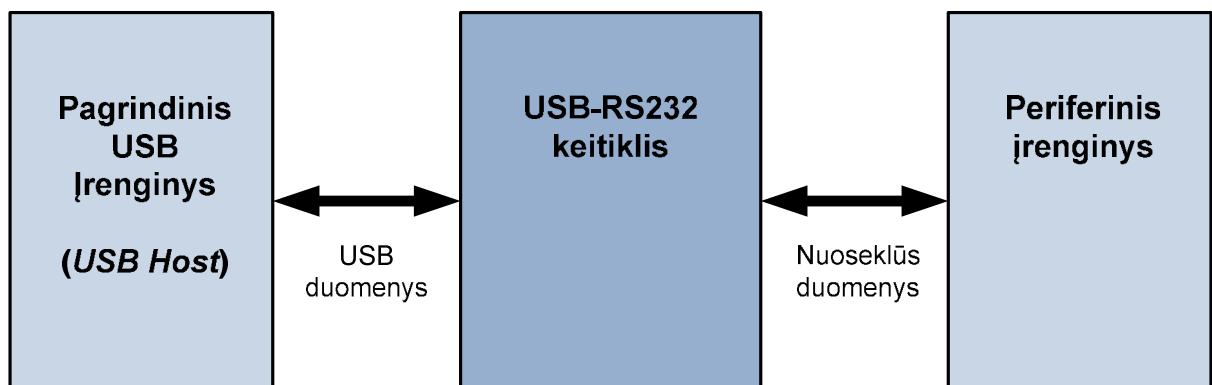
1.2. USB-UART sąsajos analizė

USB duomenų perdavimo standartas rinkoje pasirodė 1996 metais kaip naujas protokolas, palaikantis savaiminio diegimo galimybes (angl. *plug-and-play*). USB standartas leidžia vienu metu prijungti iki 127 įrenginių. Šiuo metu tai populiariausia duomenų perdavimo tarp kompiuterio ir periferinių įrenginių sąsaja buitiniuose įrenginiuose, daugiausiai kompiuteriuose. USB standartas leidžia perduoti duomenis iki 12 Mb/s greičiu. Šiuo metu labiausiai paplitę yra USB 2.0 bei USB 3.0 standartai. USB 3.0 rinkoje buvo pristatytas 2008 metais, tačiau savo pirmtako šiam standartui aplenkti nepavyko. USB 2.0 protokolas leidžia duomenis perduoti iki 480 Mb per sekundę greičiu [13]. USB 3.0 yra 10 kartų spartesnė [14], tačiau ne visuomet tokia didelė sparta yra reikalinga, kadangi periferiniai įrenginiai dažnai negali apdoroti tokiu greičiu gaunamų duomenų. Dėl sudėtingo duomenų perdavimo protokolo, USB protokolui įgyvendinti visuomet yra naudojamas su speciali tvarkyklė, kuri aprašo OS būtinus atlikti veiksmus.

Labiausiai paplitęs yra USB 2.0 standartas. Nepaisant to, USB-UART sąsaja labai dažnai naudojama tiek buitinėje tiek pramoninėje technikoje, kur nereikia perduoti labai didelių duomenų srautų. USB-UART sąsaja puikiai tinka bendravimui su periferiniais įrenginiais [15]. Todėl duomenų keitimuisi tarp LPLM ir kompiuterio ši sąsaja yra tinkama. USB protokolo pritaikymas ir realizavimas LPLM lustuose yra sudėtingas. USB protokolo ir savaiminio diegimo realizavimas reikalauja daug nuoseklių procesų. Sudėtinga realizacija lemia tai, kad norint diegti USB protokolą LPLM luste optimaliau yra įsigyti ir pritaikyti naudojimui IN modulius, nei projektuoti USB protokolą apdorojančias grandines savarankiškai. Situacija yra visiškai priešinga, kai kalbama apie nuoseklią sąsają naudojant USB-UART duomenų siuntimo ir priėmimo modulius, kurie yra nesudėtingai pritaikomi.

Dar vienas funkcinis skirtumas tarp USB ir USB-UART sąsajų yra tinkamų tvarkyklių parinkimas arba sukūrimas. Tvarkyklių suderinamumo problemos paprastai nepasireiškia masiniuose produktuose, tačiau kuriant specializuotus LPLM lustus, tvarkyklių kūrimas gali tapti gana opia problema [16]. USB standartas iš rinkos faktiškai išstūmė RS232-UART sąsajos standartą, tačiau RS232-USB arba dar dažniau naudojama USB-UART sąsaja vis dar išlieka populiari. Tokį populiarumą lemia nesudėtingas protokolo realizavimas ir paties protokolo paprastumas. Todėl USB-UART sąsaja yra tinkama ekonominės klasės produktams, kurių gamyboje labai svarbi kūrimo trukmė, kadangi tai lemia kainą. Be to, dėl duomenų apdorojimo aparatinėje įrangoje paprastumo, toks modulis sunaudoja labai nedaug programuojamo lusto resursų. Aišku, jeigu reikalinga perduoti didelius duomenų srautus arba reikalinga itin didelė duomenų perdavimo sparta, USB sąsaja yra nepakeičiama, tačiau J. Norhuzaimin'as ir H. H. Maimun'as parodė, kad galima sukurti efektyvią didelės veikimo spartos USB-UART architektūrą [17].

Vienas iš komunikacijos su kompiuteriu per USB sąsają būdų yra duomenų perdavimas naudojant USB-RS232 (USB-UART) keitiklius. Duomenų perdavimo blokinė schema pavaizduota 6 paveiksle. Keitiklio naudojimas leidžia prie kompiuterio jungti išorinius įrenginius nuoseklia duomenų perdavimo sąsaja, tačiau išnaudojant USB teikiamus privalumus [16].

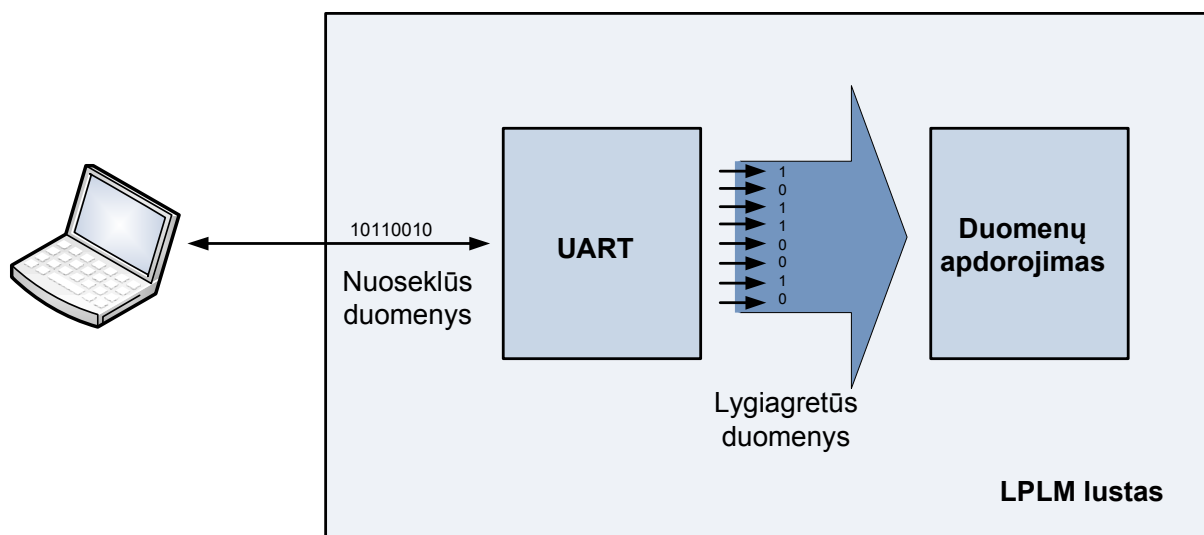


6 pav. USB-RS232 keitiklio veikimo schema – keitiklis realizuotas už LPLM lusto

Keitiklis gali būti realizuotas tiek naudojant programines tiek aparatinės priemones. Naudojant programines priemones keitiklio logika realizuojama LPLM luste, tuo tarpu naudojant aparatinės priemones, keitiklis gali būti SPIG, pavyzdžiui FT232R lustas, kuris yra integruotas Nexys3 plokštėje. USB-UART keitiklis priima iš kompiuterio USB protokolu siunčiamus duomenis, juos verčia nuoseklia asinchronine seka ir perduoda į periferinį įrenginį. Keitiklis atlieka USB

protokolo apdorojimo funkciją, todėl nereikia USB protokolo apdorojimo modulio kelti į FPGS lustą. Toks integruotas sprendimas yra efektyvus kainos ir realizavimo atžvilgiu.

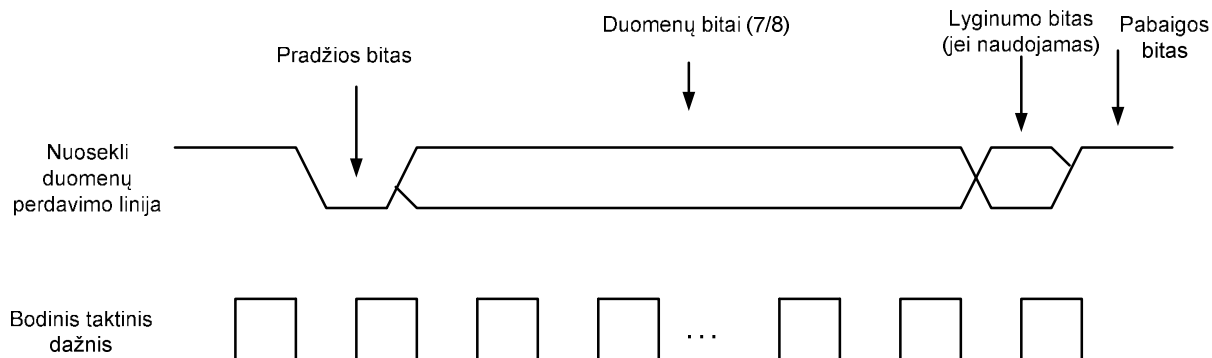
Universalus asinchroninis siųstuvas-imtuvas yra integrinė schema, naudojama asinchroniniam nuoseklių duomenų priėmimui ir perdavimui [18]. Susideda iš imtuvo ir siųstuvo. Imtuvas priima nuoseklius duomenis iš kompiuterio ir verčia juos lygiagrečiais. Siųstuvas priima lygiagrečius duomenis iš LPLM, verčia juos į nuoseklius ir perduoda kompiuteriui. Sparčiam ir sklandžiam duomenų perdavimui reikalingi buferiai. Paprastai sąsaja naudojama su RS232 standartu. Iš esmės UART sąsaja reikalinga nuosekliams duomenims versti lygiagrečiais ir perduoti kitiems LPLM luste realizuotiems blokams, pavyzdžiui, duomenų šifravimo moduliui, naudojamam koduojant duomenis. UART veikimo principas LPLM luste parodytas 7 paveiksle.



7 pav. Nuoseklių duomenų vertimas į lygiagrečius LPLM luste

FT232R yra specialios paskirties lustas, realizuojantis USB-UART sąsają. Luste yra integruotas USB protokolas, todėl nereikalinga jokia papildoma programinė ar aparatinė įranga, apdorojanti USB sąsaja gaunamus duomenis [15]. FT232 lustas su LPLM jungiamas dviem išvadais (Rx/Tx), skirtais duomenų siuntimui ir duomenų priėmimui. Nuoseklūs duomenys apdoroti UART modulio paverčiami lygiagrečiais ir duomenų magistrale gali būti perduoti apdorojimui.

Sėkmingam asinchroniniam duomenų siuntimui organizuoti į siunčiamus duomenis yra įterpiama tarnybinė informacija, skirta pranešti apie duomenų baito pradžią ir pabaigą ar duomenų lyginumą. Lyginumo informacija reikalinga aparatiniam klaidų taisymo protokolui įgyvendinti. UART sąsaja duomenys yra siunčiami baitais. Tipinis UART sąsajos duomenų siuntimo formatas parodytas 8 paveiksle.



8 pav. UART duomenų siuntimo formatas [17]

Nuoseklus duomenų srautas susideda iš šių komponentų: pradžios bitas, duomenų bitai, lyginumo bitas, pabaigos bitas. Kai ryšio linija duomenys nesiunčiami, linijoje nustatomas auštas signalo lygis. Pirmasis žemas loginis lygis yra laikomas pradžios bitu ir rodo, kad prasidėjo siuntimas. Sekantys 7-8 (atskiris atvejais gali būti 5-6) bitai yra duomenų bitai, kurie neša informaciją. Mažiausiai reikšminis bitas yra siunčiamas pirmas. Duomenų bitų skaičius nustatomas iš anksto ir turi būti vienodas abiejuose, ryšį palaikančiuose įrenginiuose. Jeigu yra nustatyta, po duomenų bitų siunčiamas lyginumo kontrolės bitas. Jis yra neprivalomas. Baito siuntimas baigiamas pabaigos bitu – nustatomas aukštas lygis ryšio linijoje.

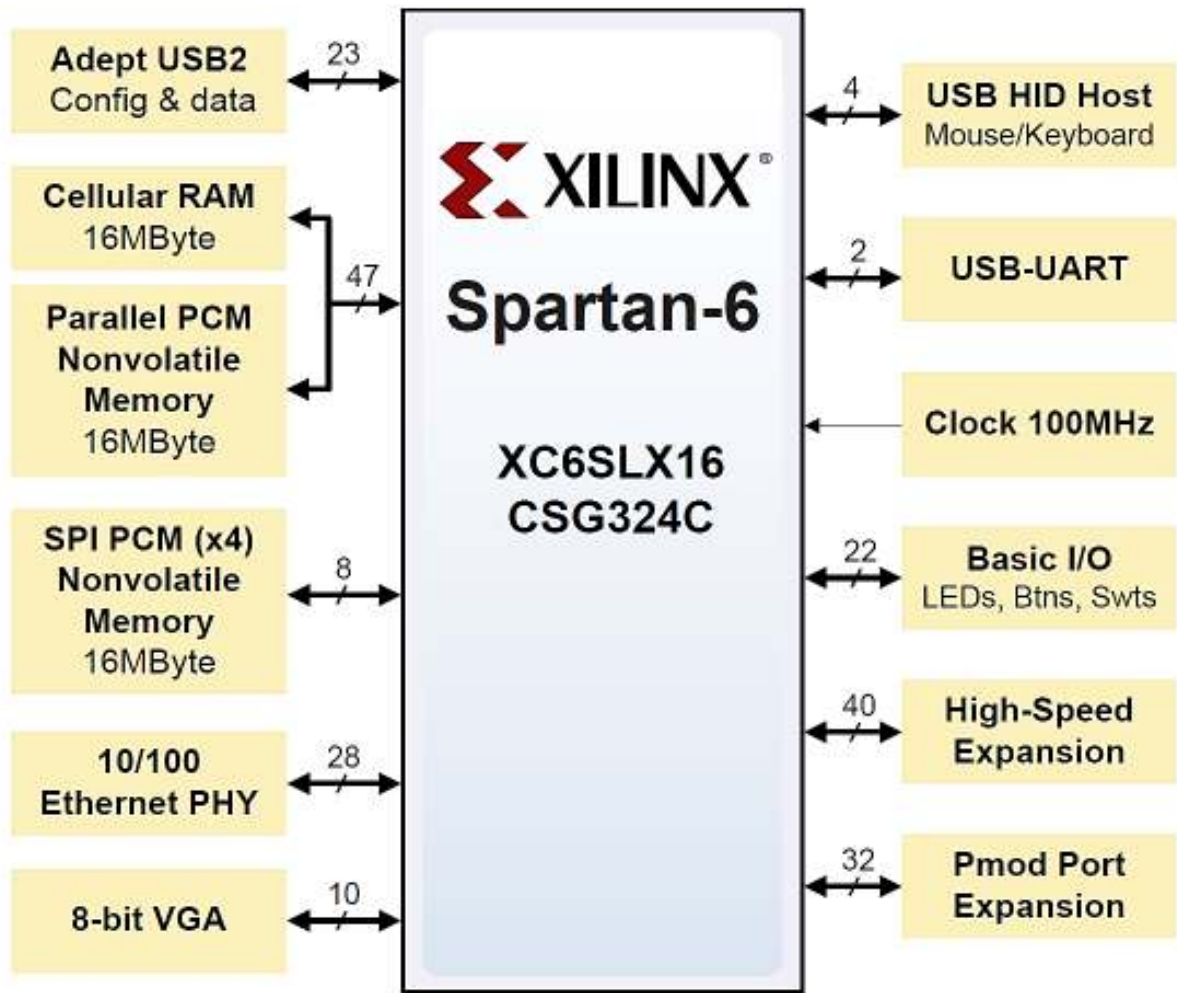
2. USB SĄSAJOS MODULIO KŪRIMAS

2.1. Naudojama aparatinė ir programinė įranga

Nexys3 yra mokomoji plokštė, skirta darbui su *Xilinx Spartan-6* LX16 LPLM lustu. *Spartan-6* LPLM šeima yra optimizuota didelei veikimo spartai. *Spartan-6* LX16 turi 2278 skiltis, iš kurių kiekviena yra sudarytas iš 6 paieškos lentelių bei 8 trigerių. Segmentų architektūra buvo aptarta 2.1 poskyryje. LPLM lustas turi integruotą greitai veikiantį 576 kB RAM bloką, dviejų tipų laikmačius (4 DCM ir 2 PLL), 32 DSP segmentus, gali dirbti iki 500 MHz dažniu. Plokštėje yra integruotas 100 MHz vidinis oscilatorius. Didesnis taktinis dažnis gali būti pasiekiamas naudojant PLL ir DCM komponentus. Kartu su *Spartan-6* LPLM lustu *Nexys3* plokštėje yra integruota nemažai periferinių įrenginių. Plokštėje yra integruota 32 MB naujo tipo *Micron* atmintis, 10/100 *Ethernet* fizinį lygmenį apdorojantis lustas, 16 MB pseudo statinė *Cellular* RAM tipo atmintis, USB-UART sąsają realizuojantis keitiklis, USB HID tipo jungtis klaviatūrai arba pelei, didelės spartos praplėtimo jungtis [19]. LPLM lusto talpa bei įvairaus tipo periferinė įranga leidžia pritaikyti *Nexys3* plokštę įvairiems įterptinių sistemų projektams, tokiems kaip *Xilinx MicroBlaze* branduolio realizavimas, IN blokų kūrimas, kriptografinės operacijos.

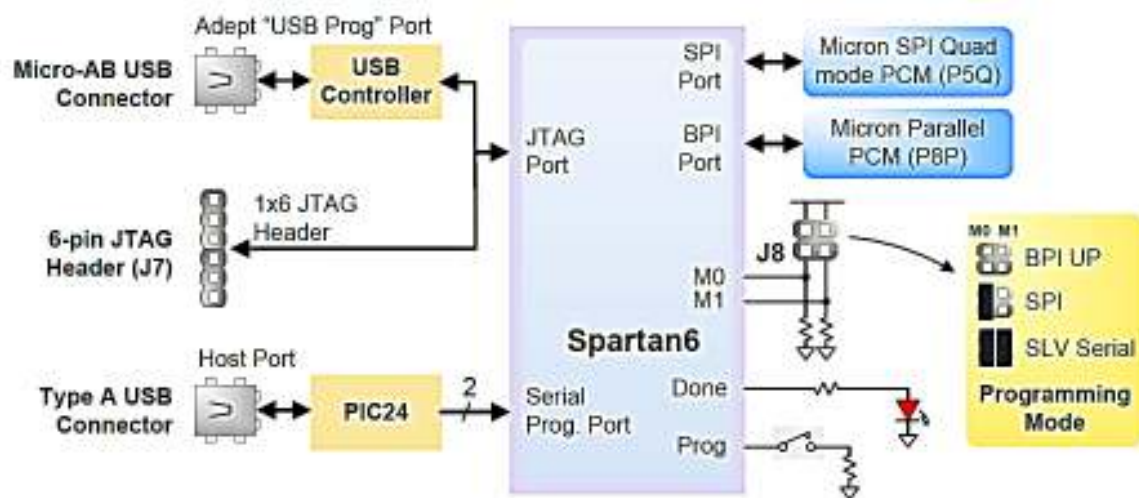
Nexys3 plokštė yra suderinama su bet kuria kompanijos *Xilinx* išleista projektavimui skirta programine įranga, tokia kaip *ChipScope*, *EDK*, *WebPack*. Be to, plokštė gali būti naudojama su *Digilent* kompanijos išleista *Adept* programine įranga, dirbančia per USB prievadą ir leidžiančia užprogramuoti LPLM struktūrą bei įkelti duomenis į ROM tipo atmintį. *Adept* programinė įranga taip pat siūlo automatizuotus plokštės patikros testus, virtualią įvesties bei išvesties patikrą, supaprastintą vartotojo duomenų perdavimo sąsają. *Nexys3* plokštės elementų blokinė diagrama pavaizduota 9 paveiksle. Plokštės pasirinkimą magistro darbo atlikimui lėmė dvi pagrindinės priežastys: integruota USB-UART sąsaja, galimybė didinti taktinį dažnį iki 500 MHz.

Nexys3 plokštė gali būti konfigūruojama 4 būdais [19]. Konfigūracinį failą galima įrašyti naudojantis *Adept* programine įranga, prijungus plokštę prie kompiuterio per specialiai programavimui skirtą USB jungtį. Konfigūracinius duomenis galima įkelti bet kuriuo metu. Konfigūracinis failas, patalpintas pastoviojoje PCM atmintyje į LPLM lustą perduodamas naudojantis BPI sąsaja; tuo tarpu failas patalpintas nuosekliojoje atmintyje perduodamas SPI sąsaja. Taip pat konfigūracinis failas gali būti nuskaitytas iš USB atminuko, prijungto prie USB HID jungties. Plokštėje yra trumpikliai, leidžiantys pasirinkti norimą konfigūracinio failo perdavimo būdą. Taip pat yra galimybė naudoti JTAG.



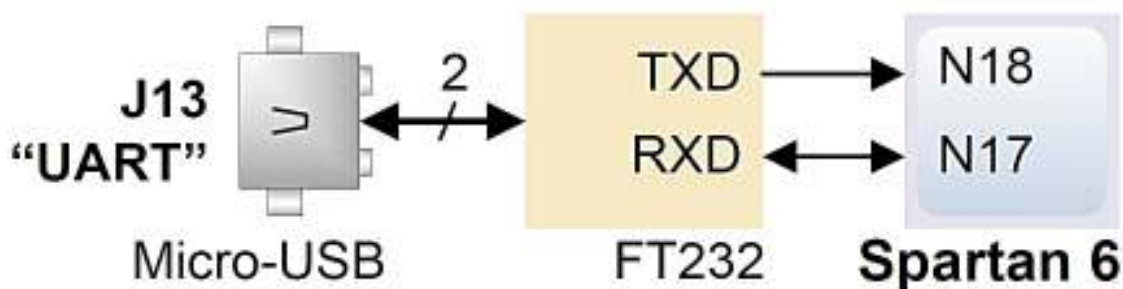
9 pav. Nexys3 plokštės blokinė schema [19]

LPLM luste konfigūraciniai duomenys yra saugomi SRAM tipo atmintyje. Konfigūracinis failas aprašo LPLM logines funkcijas ir ryšius tarp bazinių blokų. Konfigūraciniai duomenys saugomi SRAM tipo atmintyje ir išlieka iki tol, kol įrašomi nauji duomenys, sistema perkraunama arba išjungiamas maitinimas. Konfigūraciniai duomenys perduodami JTAG sąsaja gali būti .bin arba .svf tipo, tuo tarpu duomenys perduodami iš USB atmintuko, BPI arba SPI sąsaja gali būti .bit, .bin arba .msc tipo. *Xilinx* siūlomas programinis paketas ISE/WebPack arba EDK gali sukurti .bit, .svf arba .msc tipo bylas iš VHDL bei *Verilog* kodo arba iš loginių elementų schemos. *Nexys3* programavimo ir maitinimo schema pavaizduota 10 paveiksle.



10 pav. Nexys3 plokštės programavimo schema [19]

Nexys3 plokštėje USB-UART sąsaja realizuota naudojant FTDI FT232R specialios paskirties integrinį grandyną [19]. Šis lustas leidžia užmegzti komunikacijos kanalą su kompiuteriu naudojant standartinės *Windows* OS COM prievado komandas. FTDI kompanija lusto naudotojams siūlo specialias USB-COM tvarkykles, suformuojančias virtualų COM privadą ir konvertuojančias USB duomenų paketus į nuoseklius UART sąsaja perduodamus duomenis. Fizinis ryšio kanalas su LPLM suformuotas naudojant nuoseklų dviejų komunikacijos linijų (TXD/RXD) privadą. Įdiegus į kompiuterį reikiamas tvarkykles, duomenys ir valdymo komandos, siunčiamos iš kompiuterio per virtualų COM privadą į LPLM lustą patenka per N17 ir N18 lusto išvadus kaip nuoseklūs duomenys. Komunikacijos schema pateikiama 11 paveiksle.



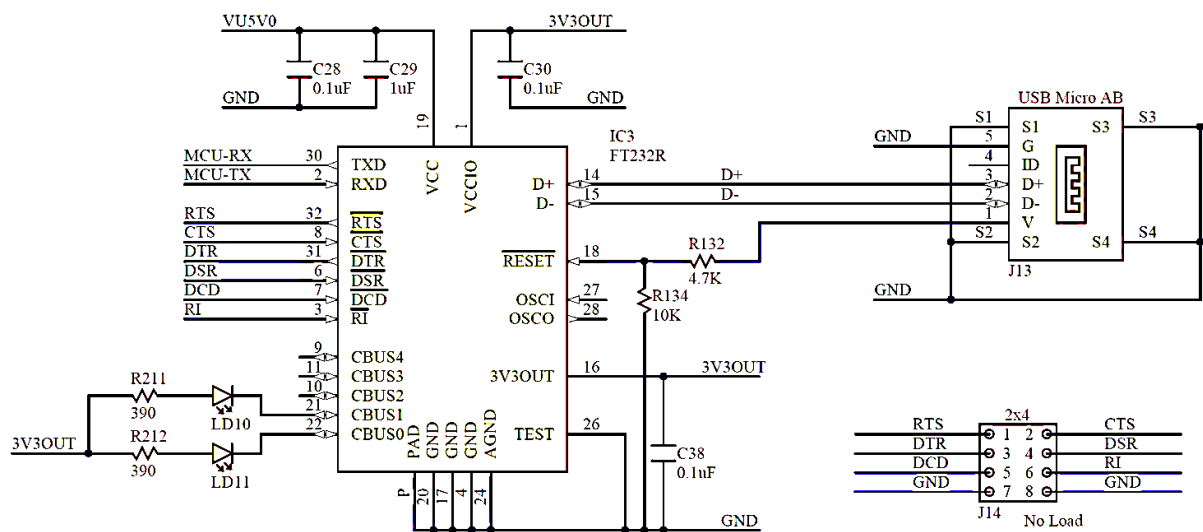
11 pav. Nexys3 plokštės ir kompiuterio sujungimo schema [19]

Iprasta, kad USB-UART sąsaja dėl nesudėtingos realizacijos naudojama ryšiui su mikrovaldikliais palaikyti. Dokumentacijoje pateikiama rekomenduojama USB-UART jungimo schema mikrovaldikliui parodyta 12 paveiksle.

[illegible]

LPLM gali dirbti daug aukštesniais dažniais nei mikrovaldikliai. *Nexys3* plokštėje yra integruotas 100 MHz oscilatorius. Naudojant fazės derinimo kilpas (PLL) ir dinaminį dažnio

keitiklį (DCM), lusto taktinį dažnį galima pakelti iki 500 MHz. Dėl to galima tikėtis pasiekti didesnę duomenų perdavimo spartą, nei įprasta. Kompiuterio ir plokštės su LPLM lustu jungimo schema parodyta 13 paveiksle. Pagrindinis skirtumas lyginant su rekomenduojamu sujungimu mikrovaldikliams yra tas, kad fizinei sąsajai sudaryti naudojami tik du išvadai. Schemoje jie pažymėti MCU-RX ir MCU-TX. Originalioje LPLM plokštėje *Nexys3* galimybė realizuoti aparatinį ryšio sudarymo patvirtinimą nenumatyta. Pagalbiniai signalai RTS, DTR, DCD, CTS, DSR ir RI yra išvesti ir numatyta vieta papildomai jungčiai. Duomenų perdavimo indikacijai naudojami CBUS1 ir CBUS2 išvadai. Jie sujungti su šviesos diodais (LED) ir parodo, ar linija siunčiami duomenys.



13 pav. *Nexys3* plokštės ir kompiuterio jungimo schema [20]

FT232R lusto dokumentacijoje numatyta, galimybė naudoti išorinį osciliatorių. *Nexys3* plokštėje ši savybė neišnaudota, ir naudojamas tik vidinis FT232R lusto 12 MHz osciliatorius. Tuo tarpu *Nexys3* plokštėje yra integruotas 100 MHz osciliatorius. FT232R luste naudojamas bodinio dažnio generatorius, lemiantis UART sąsajos bodinę spartą, kuri gali būti nuo 183 bodų iki 3000000 bodų. Lustas gali dirbti bet kuria standartine bodine sparta, esančią šiame intervale. Taip pat galima naudoti nestandartinę spartą, kuri apskaičiuojama pagal formulę:

$$\text{Bodinė sparta} = \frac{3000000}{n+x};$$

Čia n – sveikasis skaičius esantis intervale nuo 2 iki 2^{14} (16384), x – dedamoji, galinti įgyti reikšmes 0, 0,125, 0,25, 0,375, 0,5, 0,625, 0,75 arba 0,875. Kai $n = 1$, $x = 0$: bodinės spartos daliktis negali įgyti reikšmių intervale nuo 1 iki 2 [8]. Naudoti nestandartinę bodinę spartą yra nesudėtinga. Pakanka terminalinėje programoje nurodyti norimą dydį. FTDI tvarkyklė pagal norimą gauti vertę apskaičiuoja reikiamus daliklius.

2.2. USB-UART modulio LPLM luste kūrimas

USB-UART sąsajos pasirinkimą komunikacijai tarp LPLM lusto lėmė naudojama aparatinė įranga. USB-RS232 keitiklis, integruotas Nexys3 mokomojoje plokštėje sujungtas pagal schemą, pavaizduotą 13 paveiksle. FTDI DT232 USB-UART tiltas leidžia perduoti duomenis tarp LPLM plokštės ir kompiuterio per standartinę virtualią COM sąsają. Duomenys siunčiami nuoseklia laidų pora (RxD/TxD). Fiziškai su kompiuteriu plokštė sujungta naudojant fizinį USB prievadą.

USB-UART modulis be duomenų kaupimo (buferių) yra pats paprasčiausias USB sąsajos palaikymo modulio variantas. Duomenys siunčiami į LPLM lustą nuoseklia sąsaja. UART imtuvas juos apdoroja ir paverčia lygiagrečiu duomenų srautu. UART siųstuvas, realizuotas LPLM luste priima lygiagrečius duomenis, juos verčia nuosekliais duomenimis ir išsiunčia į kompiuterį. Tokį USB-UART modulį sudaro 3 pagrindiniai komponentai: UART siųstuvas, UART imtuvas ir dažnio generatorius. Dažnio generatorius, priklausomai nuo duomenų siuntimo bodinės spartos, generuoja signalą, kuomet reikia nuskaityti arba išsiųsti priimamą duomenų bitą. UART siųstuvas ir UART imtuvas realizuotas remiantis pavyzdžiais. UART sąsajos veikimo testavimui reikalinga duomenis tiek siųsti į LPLM, tiek juos priimti. Dėl to taikoma nesudėtinga grįžamojo ryšio schema. Imtuvo išėjimas sujungiamas su siųstuvo įėjimu. Taip priimti duomenys iš karto patenka į siųstuvą ir yra priimami kompiuterio.

UART modulį sudaro du pagrindiniai blokai: duomenų siuntimo ir duomenų priėmimo. Imtuvas priima duomenis iš kompiuterio, o siųstuvas tuo tarpu siunčia duomenis į kompiuterį. Taip pat naudojamas bodinio dažnio generatorius, kuris užtikrina savalaikį duomenų nuskaitymą. Tam naudojama signalo diskretizavimo aukštesniu dažniu (angl. *oversampling*) procedūra. Diskretizavimo dažnis yra 16 kartų didesnis už bodinę spartą, tai reiškia kad kiekvienas bito ilgis yra 16 generatoriaus taktų.

Bodinio dažnio generatorius yra reikalingas nustatyti kiekvieno bito (arba bodo) išsiuntimo laiką. Kadangi duomenų siuntimas yra asinchroninis, reikia užtikrinti savalaikį duomenų nuskaitymą. Dėl to yra generuojamas signalas, kurio dažnis yra 16 kartų didesnis už reikiamą bodinę spartą. Tam, kad nebūtų pažeistas sinchroniškumas yra generuojamas aukšto lygio signalas,

nurodantis, kad jau reikia nuskaityti duomenis. Generatoriaus skaitiklio vertė apskaičiuojama pagal formulę:

$$\text{Skaitiklio vertė} = \frac{\text{LPLM veikimo dažnis}}{d \cdot \text{bodinė sparta}} \quad [21] \quad (1)$$

Vardiklyje esanti konstanta d rodo, kiek kartų atskaitų generatoriaus dažnis yra didesnis už bodinę spartą. Kaip jau buvo minėta, labiausiai paplitusi vertė yra 16. 3 lentelėje pateikiamos apskaičiuotos skaitiklio vertės kiekvienam naudojamam bodiniam dažniui. Be to, lentelėje pateikiama paklaida, atsirandanti dėl skaitiklio reikšmės apvalinimo. Palyginimui pateikiama FT232R lusto paklaidos. Duomenys gaunami apskaičiuojant artimiausius naudojamai bodinei spartai koeficientus ir surandant tikrąjį bodinį dažnį. Šios paklaidos lemia duomenų nuskaitymo tikslumą. Tam, kad ryšio kanalas veiktų be klaidų, bendra paklaida neturėtų viršyti 3% [22]. Iš lentelėje pateikto bendros paklaidos stulpelio matyti, kad visiems bodiniams greičiams nuo 14400 bodų/s iki 460800 bodų/s paklaida neviršija 1%. Dideliems bodiniams greičiams paklaida yra 3-4%. Tikėtina, kad šias paklaidas galima būtų sumažinti pakoregavus LPLM lusto taktinį dažnį.

3 lentelė. Generatoriaus parametrų skaičiavimas

Bodinė sparta	Generatoriaus veikimo dažnis ($d = 16$)	Skaitiklio vertė, $\left(\frac{100 \text{ MHz}}{\text{Gen.veikimo dažnis}}\right)$	Paklaida, % (FPGA)	Paklaida, % FT232R	Bendra paklaida, %
14400	230400	434	0,01	0,02	0,03
19200	307200	326	0,15	0,00	0,15
38400	614400	162	0,15	0,00	0,15
57600	921600	109	0,45	0,08	0,53
115200	1843200	54	0,47	0,16	0,63
230400	3686400	27	0,47	0,16	0,63
460800	7372800	14	3,22	0,16	3,38
921600	14745600	7	3,22	0,16	3,38
1500000	22588224	4	4,00	0,00	4,00
3000000	48000000	2	4,00	0,00	4,00

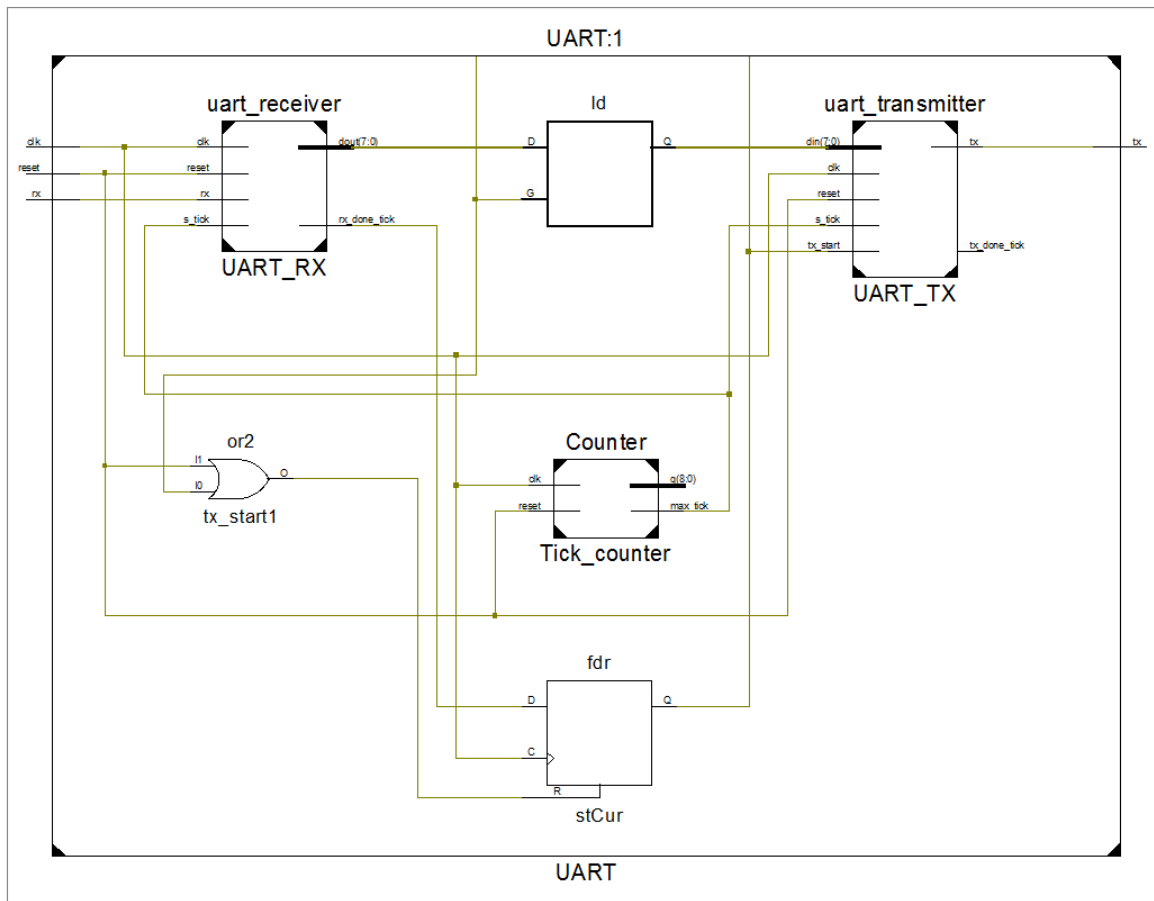
Imtuvas (*uart_receiver*) turi 4 įėjimo ir 2 išėjimo signalus. Imtuvo VHDL komponentas realizuotas remiantis pavyzdžiu pateikiamu Pong P. Chu'o [21]. *Clk* ir *Reset* signalai yra bendri visai grandinei. Šie signalai yra taktinio dažnio ir sistemos perkrovos signalai. Be paminėtų, imtuvas turi dar 2 įėjimo signalus. *Rx* įėjimas yra sujungtas su fiziniu LPLM išvadu, kuriuo iš kompiuterio patenka nuoseklūs duomenys. *S_tick* signalas skirtas sujungti imtuvą su bodinio dažnio generatoriumi. Išėjimo signalas *dout(7:0)* išveda iš imtuvo lygiagretų duomenų baitą, kuris buvo priimtas imtuve, tuo tarpu *rx_done_tick* signalas yra informacinis ir skirtas perduoti informaciją

apie tai, kad duomenų baitas priimtas sėkmingai. Naudojamos dvi konstantos, apibrėžiančios duomenų bitų skaičių ir pabaigos bitų skaičių. Imtuvas veikia kaip būsenų automatas, turintis keturias būsenas: *idle*, *start*, *stop* ir *data*. *Idle* būsena laukia *rx* linijoje žemo lygio signalo. Signalui atsiradus pereinama į *start* būseną, laukiama 7 generatoriaus ciklus ir pereinama į *data* būseną. Šioje būsenoje laukiama 16 ciklų ir fiksuojamas pirmas duomenų bitas. Analogiškai fiksuojami ir kiti duomenys. Perskaičius visą baitą pereinama į *stop* būseną laukiama 16 taktų.

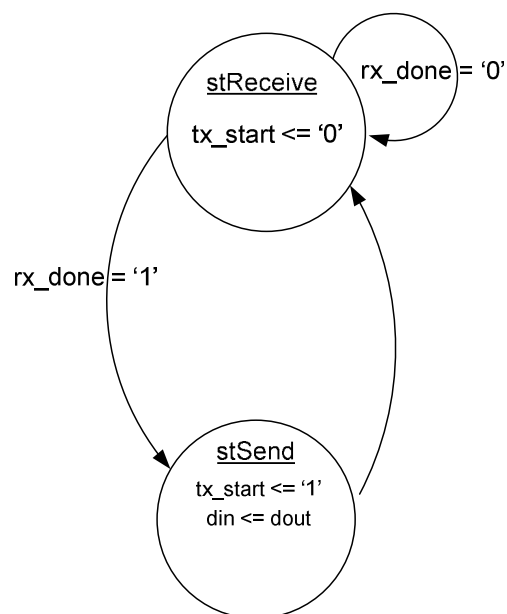
Siųstuvas taip pat turi bendrus visai grandinei signalus *clk* ir *reset*, taip pat yra naudojamas *s_tick*, bodiniam dažniui iš generatoriaus priimti. Lygiagretūs duomenys (1 baitas) į siųstuvą patenka *din(7:0)* signalu. *Tx_done* yra informacinio pobūdžio signalas, informuojantis apie sėkmingai perduotą baitą, *tx_start* paleidžia UART siųstuvą. *Tx* signalas yra sujungtas su fiziniu LPLM išvadu ir juo perduodami duomenys iš lusto į kompiuterį. Duomenų perdavimo valdymui naudojamas būsenų automatas. Automatas veikia analogiškai kaip ir imtuvo būsenų automatas, tik duomenys ne priimami, o išsiunčiami.

Tyrimams pasirinktos dviejų tipų UART modulio schemas. Pirmoji schema sudaryta iš 3 pagrindinių komponentų: bodinio dažnio generatoriaus, siųstuvo ir imtuvo. Būsenų automatas valdo duomenų perdavimą. Nei siųstuvas nei imtuvas neturi jokių atminties elementų. Naudojama atgalinio ryšio (angl. *loop back*) sujungimo schema. Duomenys, priimti UART imtuvo iš karto yra perduodami UART siųstuvui. Blokinė šios struktūros schema gauta sugeneravus ją ISE programine įranga, pavaizduota 14 paveiksle, programos VHDL kodas pateikiamas A priede. Nagrinėjamas USB-UART modulis neturi atminties elementų. Duomenų siuntimą ir priėmimą valdo baigtinis būsenų automatas. Automatą sudaro dvi būsenos: *stReceive* ir *stSend*. *stReceive* būsenoje yra laukiama kol imtuvas priims siunčiamus duomenis. Duomenų išsiuntimo signalas laikomas žemame lygyje. Kai imtuvas priima duomenų baitą, *rx_done* signalas pakeičia lygį iš žemo į aukštą. Pasikeitus šiam signalui automatas pereina į būseną *stSend*. Šioje būsenoje keičiama *tx_done* signalo būsena į aukštą lygį – tai leidimas rašyti duomenis. Taip pat imtuvo išėjimas nukreipiamas į siųstuvo įėjimą ir duomenys išsiunčiami. Automatas iš karto keičia būseną į *stReceive* ir laukia, kol bus priimtas naujas duomenų baitas.

Aptartasis UART modulis neturi atminties elementų. Dėl bodinio dažnio generavimo paklaidų ir dėl vėlinimų linijose, tikėtina, kad dalis duomenų gali būti prarasta. Dėl to buvo sudarytas USB-UART modulis, turintis atminties elementus, FIFO buferius, skirtus lygiagrečiai siųsti ir priimti duomenis. Šią funkciją puikiai atlieka FIFO (angl. First In First Out) buferis. Pirmasis į buferį patekęs baitas yra siunčiamas į FIFO išėjimą. Tokia struktūra leistų suderinti siųstuvo ir imtuvo veikimą taip, kad nebūtų prarandama dėl vėlavimų ir paklaidų.

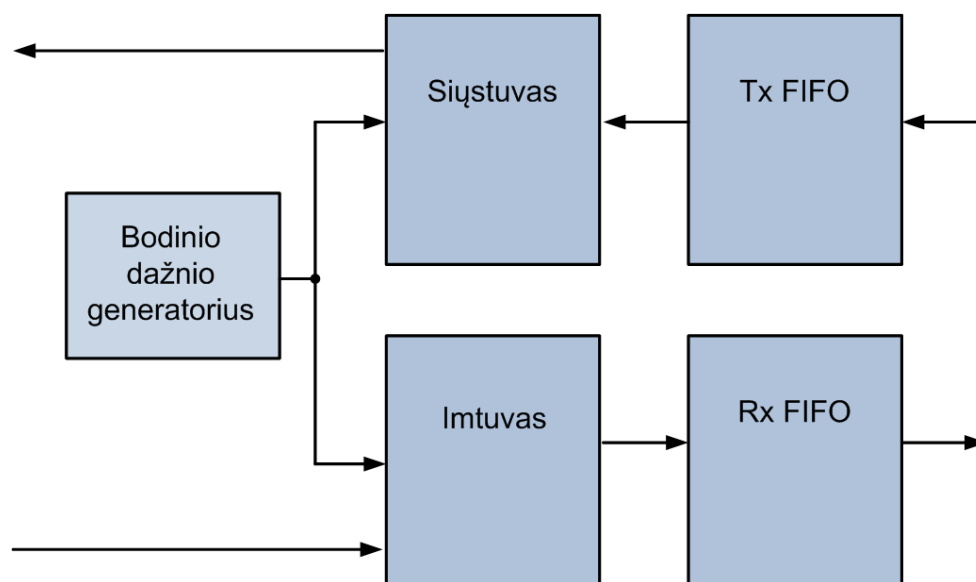


14 pav. UART modulis be atminties elementų



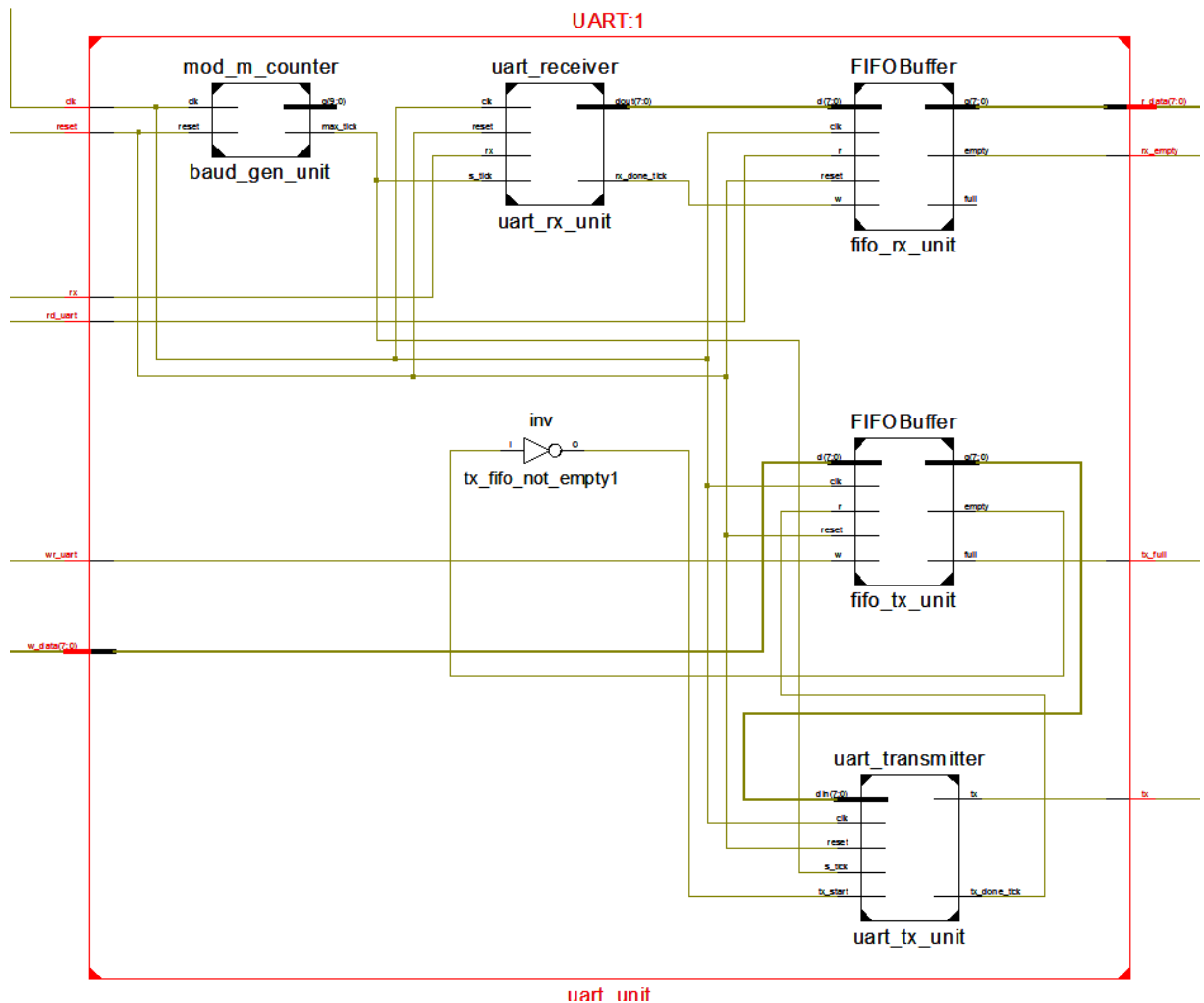
15 pav. UART valdantis būsenų automatas

USB-UART modulio su FIFO buferiais blokinė schema pavaizduota 16 paveiksle. Bodinio dažnio generatorius sujungtas su siųstuvu ir imtuvu, kaip ir prieš tai aptartoje schemoje. Prie siųstuvo bei imtuvo taip pat prijungti du 8 baitų FIFO buferiai [23], kurie saugo duomenis prieš išsiunčiant ir prieš gaunant. Pagal šią schemą suprojektuotas VHDL modulis pavaizduotas 17 paveiksle. Toks UART modulis gali sėkmingai veikti pilnai dvikrypčiu režimu (angl. *full-duplex*). Dirbant šiuo režimu duomenys siunčiami ir priimami tuo pačiu metu.



16 pav. USB-UART su FIFO buferiais blokinė schema

FIFO buferis turi po 2 informacinius signalus *empty* ir *full*, parodančius buferio būseną. Jeigu *empty* signalas yra aukštas, jis rodo, kad buferis yra visiškai tuščiau, t.y. jame nėra įrašyta duomenų. Jeigu *full* signalas yra aukštas, tai jis informuoja, kad buferis yra persipildęs ir daugiau duomenų į jį netelpa. Esant buferio persipildymui jeigu nespėjama išsiųsti arba priimti duomenų, tai dalis jų gali būti prarasta. Be to yra galimybė, jog duomenys bus neteisingai perskaityti. Taip pat buferiai turi po du valdymo signalus *r* ir *w*. Šie signalai perduoda į buferį komandas skaityti ir rašyti. Jeigu *r* signalo lygis yra aukštas, iš buferio gali būti skaitomi duomenys. Jeigu *w* signalo lygis aukštas, tuomet į buferį gali būti rašomi duomenys. Galima *r* ir *w* signalų kombinacija, kai duomenys yra ir skaitomi ir rašomi tuo pačiu metu.



17 pav. USB-UART modulio schema

FIFO buferis turi po 2 informacinius signalus *empty* ir *full*, parodančius buferio būseną. Jeigu *empty* signalas yra aukštas, jis rodo, kad buferis yra visiškai tuščias, t.y. jame nėra įrašyta duomenų. Jeigu *full* signalas yra aukštas, tai jis informuoja, kad buferis yra persipildęs ir daugiau duomenų į jį netelpa. Esant buferio persipildymui jeigu nespėjama išsiųsti arba priimti duomenų, tai dalis jų gali būti prarasta. Be to yra galimybė, jog duomenys bus neteisingai perskaityti. Taip pat buferiai turi po du valdymo signalus *r* ir *w*. Šie signalai perduoda į buferį komandas skaityti ir rašyti. Jeigu *r* signalo lygis yra aukštas, iš buferio gali būti skaitomi duomenys. Jeigu *w* signalo lygis aukštas, tuomet į buferį gali būti rašomi duomenys. Galima *r* ir *w* signalų kombinacija, kai duomenys yra ir skaitomi ir rašomi tuo pačiu metu.

Siųstuvas, imtuvas, generatorius ir buferiai sudaro UART komponentą. Šis komponentas turi tokius signalus:

Clk, reset – bendri visai grandinei taktavimo ir perkrovos signalai;

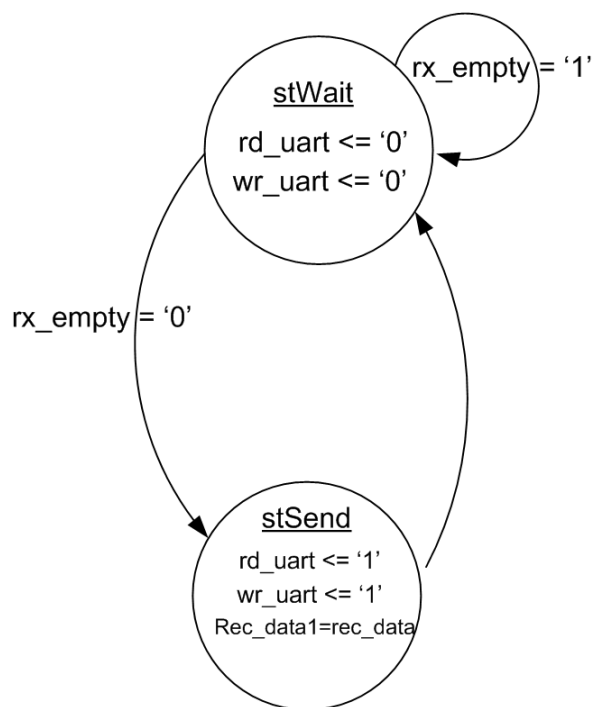
Rd_uart, wr_uart – UART modulio valdymo signalai. *Rd_uart* leidžia skaityti imtuvo buferį, o *wr_uart* leidžia rašyti duomenis į siųstuvo buferį;

Rx, tx – duomenų perdavimo nuoseklia sąsaja signalai, sujungti su LPLM išvadais, kurie savo ruožtu sujungti su RT232R išvadais;

Tx_full, rx_empty – informacinio pobūdžio signalai informuojantys apie siųstuvo buferio persipildymą ir duomenų nebuvimą imtuvo buferyje;

W_data, r_data – duomenų magistralės. *W_data* yra lygiagrečiai perduodami duomenys į siųstuva, *r_data* – imtuvo priimti lygiagretūs duomenys.

UART sąsajos patikrinimui sudaryta atgalinio ryšio schema. Atgalinio ryšio duomenų perdavimo sistemos testavimo kodas pateiktas B priede. Duomenys patenkantys į *r_data* magistralę iš karto perduodami į *w_data* išsiuntimui. Buferių tikrinimą ir duomenų perdavimą LPLM lusto viduje kontroliuoja būsenų automatas. Būsenų automato schema pateikiama paveiksle 18 paveiksle.

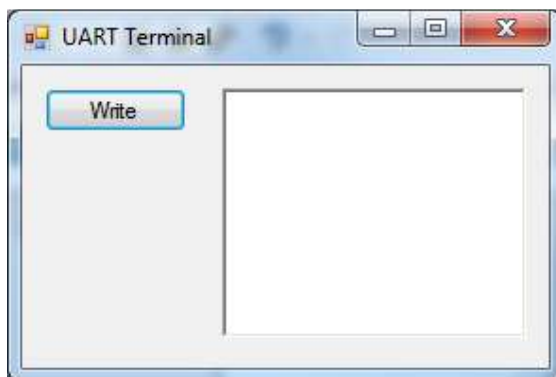


18 pav. UART su FIFO būsenų diagram

Baigtinį būsenų automata sudaro dvi būsenos *stSend* ir *stWait*. Iš *stWait* būsenos automas pereina į kitą būseną kai pasikeičia *rx_empty* signalo loginis lygis iš aukšto į žemą. Tai reiškia, kad imtuvo buferis yra ne tuščias. Pasikeitus loginiam lygiui yra pereinama į *stSend* būseną. Šioje būsenoje suteikiami leidimai rašyti duomenis į siųstuvo FIFO buferio ir skaityti duomenis iš imtuvo FIFO buferio. Siųstuvui perduodamas atmintyje esantis duomenų baitas. Atlikus šiuos veiksmus būsenų automatas grįžta į būseną *stWait* ir yra laukiama, kol buferyje bus duomenų.

2.3. Komunikacijos programos kompiuterio OS kūrimas

Komunikacijos programa ryšiui tarp kompiuterio ir LPLM lusto palaikyti kuriama *Microsoft Visual Studio 2012* programine įranga. Naudojama C# programavimo kalba. Sukuriamas pagrindinis programos langas, kuriame numatyti du valdymo mygtukas *Write*. Taip pat realizuotas duomenų išvedimo laukas. Pagrindinio programos lango iliustracija pateikiama 19 paveiksle.



19 pav. Terminalo langas

Mygtukas *Write* yra skirtas duomenų siuntimui pradėti. Paspaudus *Write* mygtuką yra kreipiamasi *ButtonClick* metodu, kuriame yra užmezgamas ryšys su virtualiu COM prievadu. Programoje naudojama standartinė .NET Framework 4.5 klasė *SerialPort*, skirta nuosekliam duomenų perdavimui. Ši klasė turi visus pagrindinius metodus, reikalingus ryšio kanalui sudaryti. Naudojantis klasėje aprašytomis funkcijomis ir sąsajos savybėmis yra aprašomi USB-UART sąsajos parametrai:

```
serialPort1.PortName = "COM1";  
serialPort1.BaudRate = 14400;  
serialPort1.DataBits = 8;  
serialPort1.Parity = Parity.None;  
serialPort1.StopBits = StopBits.One;  
serialPort1.Handshake = Handshake.None;
```

Patikrinama, ar virtualus COM prievadas nėra atidarytas. Jei atidarytas, tuomet jį uždarome. Tai padės išvengti konfliktų su kitomis norinčiomis dirbti su prievadu programomis.

Vykdamas *Write* mygtuko metodą yra sudaromas 10kB duomenų siuntimo failas. Failas sudaromas naudojant atsitiktinių reikšmių generatorių, kuris iš duotos ASCII simbolių eilutės atsitiktinai išrenka norimą kiekį simbolių ir surašo juos į simbolių masyvą:

```
int size = 10240;
var chars =
"abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789";
var random = new Random();
var result = new string(Enumerable.Repeat(chars, size).Select(s =>
s[random.Next(s.Length)]).ToArray());
```

Paruošus duomenis siuntimui yra atidaromas COM prievadas ir vykdomas duomenų siuntimas bei priėmimas:

```
serialPort1.Open();
richTextBox1.AppendText("open");
TimeCounter.Start();
serialPort1.Write(result);
serialPort1.Read(read_buffer, 0, size);
TimeCounter.Stop();
```

Pirmiausiai atidaromas virtualus COM prievadas, paleidžiamas laikmatis, skaičiuojantis kiek laiko kompiuteris vykdo operacijas. Duomenys siunčiami ir rašomi į nuoseklų prievadą. Pasibaigus visoms operacijoms laikmatis stabdomas. Pasibaigus duomenų siuntimui prievadas yra uždaromas. Priimti duomenys įrašomi į tekstinę bylą. Tarnybinė informacija, apie atidarytą ir uždarytą prievadą bei apie duomenų skaitymo/rašymo trukmę išvedama į tam skirtą terminalo lauką. Visą programos kodą galima rasti C priede.

Minimalius poreikius vykdoma terminalo programa geba siųsti ir priimti duomenis iš LPLM lusto. Eksperimentuojant su programa pastebėta, kad ji negali dirbti pilnai dvikrypčiu režimu. Tam reikia lygiagretinti vykdomas funkcijas. Dėl to, kad terminalas nedirba pilnai dvikrypčiu duomenų perdavimo režimu yra prarandama dalis informacijos. Terminalas geba tik siųsti duomenis į LPLM lustą arba tik juos priimti. Jis tinka duomenims į LMLM siųsti, kai luste veikianti sistema juos grąžina ne iš karto, t. y. duomenys iš pradžių yra apdorojami, o po to grąžinami atgal į kompiuterį.

3. DUOMENŲ PERDAVIMO TARP LPLM IR KOMPIUTERIO TYRIMAS

Magistro darbo metu siekiama ištirti USB-UART sąsajos panaudojimo ryšiui tarp LPLM ir kompiuterio palaikyti. Pirmajame tyrimo etape nustatoma USB-UART ryšio kokybė. Pagrindinis kokybę rodantis parametras – klaidų kiekis perduodant duomenis. Klaidų kiekis skaičiuojamas abiem komunikacijoms moduliams: baziniams UART moduliui be priimtų duomenų saugojimo ir moduliui su FIFO buferiu, kuriame kaupiami priimti duomenys. Nustačius klaidų kiekį yra įvertinamas optimalus duomenų srauto dydis. Optimalus duomenų srauto dydis nustatomas analizuojant skirtumus tarp išsiųstų ir priimtų duomenų. Fiksuojama pirmos klaidos atsiradimo vieta. Žinant optimalų duomenų paketo dydį kiekvienam bodiniam greičiui, galima optimizuoti duomenų perdavimo protokolą siekiant išlaikyti maksimalią perdavimo spartą ir kokybę.

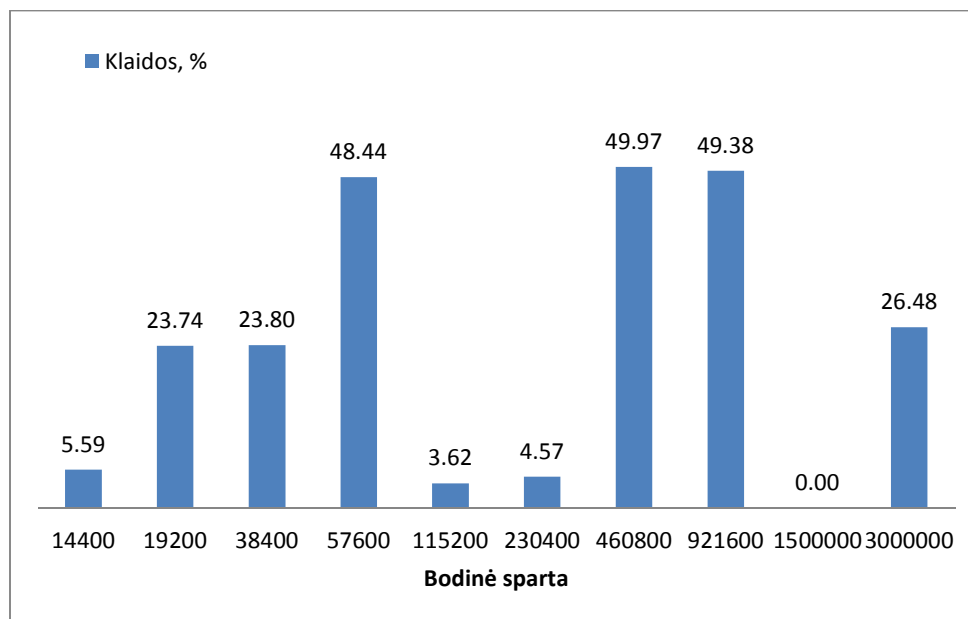
3.1. Ryšio kokybės nustatymas

Nustatant ryšio kokybę į LPLM iš kompiuterio buvo siunčiama tekstinė 10 kB dydžio byla. Kadangi UART modulis sujungtas pagal atgalinio ryšio schemą, duomenys priimti UART imtuvo iš karto buvo siunčiami atgal į UART siųstuvą. Kompiuterio priimti duomenys buvo įrašomi į tekstinę bylą. Gauti duomenys analizuojami ir lyginami su išsiųstais duomenimis: skaičiuojamas negautų baitų skaičius ir sugadinti baitai. 4 lentelėje pateikiami gauti rezultatai. Duomenų analizė atliekama naudojant *UltraCompare Professional* programinę įrangą. Naudojamas *Binary (smart)* palyginimo algoritmas, kuris identifikuoja sutampančius blokus ir baitų poslinkius. Šis algoritmas idealiai tinka, kai yra prarandama duomenų arba jie yra įterpiami [24].

4 lentelė. Duomenų perdavimo kokybė naudojant USB-UART modulį be buferio LPLM luste

Bodinis greitis, bodai/s	Priimta baitų (iš 10 kB)	Sugadinta baitų	Nepriimti duomenys, %	Sugadinti duomenys, %	Iš viso klaidų, %
14400	9668	0	5,59	0,00	5,59
19200	7809	0	23,74	0,00	23,74
38400	7803	0	23,80	0,00	23,80
57600	5280	0	48,44	0,00	48,44
115200	9869	0	3,62	0,00	3,62
230400	9772	0	4,57	0,00	4,57
460800	5163	40	49,58	0,39	39,97
921600	5183	0	49,38	0,00	49,38
1500000	10240	0	0,00	0,00	0,00
3000000	7528	2608	1,02	25,47	26,48

Kaip matyti iš gautų duomenų, geriausiai duomenys perduodami 1500000 bodų per sekundę greičiu. Išsiųsti ir priimti duomenys idealiai sutapo. Duomenų nebuvo nei prarasta, nei sugadinta. Taip pat geri rezultatai perduodant duomenis 14400, 115200, 230400 bodine sparta. Klaidų skaičius yra atitinkamai 5,59 %, 3,62 % ir 4,57%. Duomenų perdavimo kokybė kitais greičiais gana prasta: klaidų aptikta nuo 23,74 % iki 49,38 %. 20 paveiksle pavaizduotoje diagramoje aiškiai matomas skirtumas tarp sklandžiausiai ir prasčiausiai perduotų duomenų.



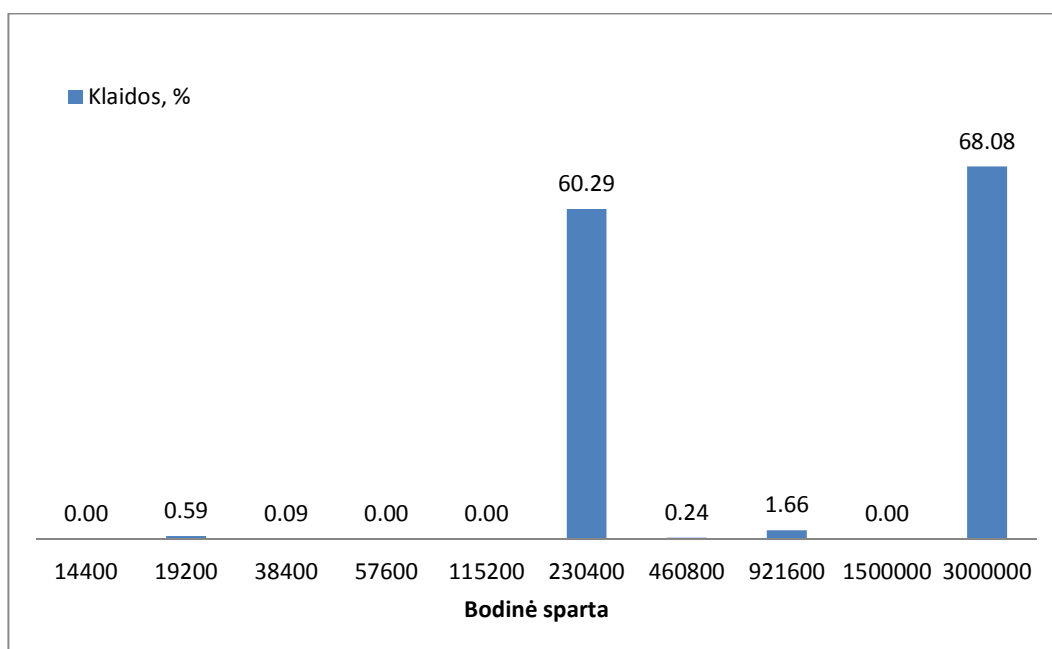
20 pav. Duomenų siuntimo UART moduliui be FIFO rezultatai

Realizavus USB-UART modulį su FIFO buferiais buvo atliktas analogiškas tyrimas. Taip pat buvo siunčiamas 10 kB duomenų blokas. Gauti rezultatai pateikiami 5 lentelėje.

5 lentelė. Duomenų perdavimo kokybė naudojant USB-UART modulį su FIFO buferiais

Bodinis greitis, bodai/s	Priimta baitų (iš 10 kB)	Sugadinta baitų	Nepriimti duomenys, %	Sugadinti duomenys, %	Iš viso klaidų, %
14400	10240	0	0,00	0,00	0,00
19200	10231	51	0,09	0,50	0,59
38400	10231	0	0,09	0,00	0,09
57600	10240	0	0,00	0,00	0,00
115200	10240	0	0,00	0,00	0,00
230400	10286	6128	-0,45	59,84	60,29
460800	10215	0	0,24	0,00	0,24
921600	10238	168	0,02	1,64	1,66
1500000	10240	0	0,00	0,00	0,00
3000000	10441	6800	-1,67	66,74	68,08

Nagrinėjant 5 lentelėje pateiktus duomenis pastebima, kad siunčiamų duomenų praradimas ženkliai sumažėjo, tačiau atsirado kita problema – siunčiant duomenis 230400 ir 3000000 bodine sparta, duomenų buvo priimta daugiau nei išsiųsta. Lentelėje šiuos atvejus žymi minuso ženklas. Skaičiuojant bendrą klaidų procentą, papildomai atsiradę baitai pridedami prie sugadintų baitų skaičiaus. Akivaizdus duomenų perdavimo pagerėjimas matomas iš to, kad net 4 atvejais nebuvo prarasta ir sugadinta duomenų. Siunčiant duomenis 14400, 57600, 115200 ir 1500000 bodine sparta, grįžę duomenys yra identiški išsiųstiesiems. Dviem atvejais (230400 bodų/s ir 3000000 bodų/s) ne tik buvo priimta daugiau simbolių nei išsiųsta, bet taip pat buvo sugadinta labai daug duomenų – 60,29 % ir 68,08 %. Rezultatų diagrama pateikiama 21 paveiksle.



21 pav. Duomenų siuntimo UART moduli su FIFO buferiu rezultatai

Siekiant palyginti abiejų tyrimų rezultatus sudaryta 22 paveiksle pavaizduota diagrama. Diagramoje sugretinami abiejų eksperimentų metu gauti rezultatai. Pastebima, kad siunčiant duomenis abiem moduliais, 1500000 bodų/s sekundę duomenų perdavimo sparta, neaptikta klaidų nei prarandant siunčiamus simbolius, nei sugadinant duomenis. Duomenų perdavimo rezultatai ženkliai pagerėjo ir visais kitais atvejais, išskyrus duomenų siuntimą 230400 bodų/s ir 3000000 bodų/s sparta. Iš esmės, galima daryti išvadą, kad sukurtas USB-UART modulis su FIFO buferiu tinka sparčiam duomenų perdavimui. Tinkamiausia duomenų perdavimo sparta – 1500000 bodų/s. Šiuo atveju siunčiamas duomenų paketas susideda iš 2 tarnybinių bitų (pradžio ir pabaigos) ir 8

duomenų bitų. Siunčiant 1500000 bodų/s greičiu, galima pasiekti 1,2 Mb/s efektyvią duomenų perdavimo spartą, kuri yra tinkama greitam duomenų perdavimui tarp LPLM lusto ir kompiuterio.

3.2. USB-UART modulių palyginimas

VHDL projektavimo kalba sudaryti USB-UART moduliai naudoja nedaug LPLM resursų. 6 lentelėje pateikiamas komunikacijos modulio naudojamų elementų palyginimas su lusto turimais elementais.

6 lentelė. Naudojamų resursų palyginimas

	XC6SLX16	UART be FIFO		UART su FIFO	
	Iš viso komponentų	Panaudota komponentų	Procentas	Panaudota komponentų	Procentas
Sekcijų skaičius	2278	46	2 %	114	5 %
PL skaičius	9112	120	1 %	241	2 %
Trigerių skaičius	18224	57	1 %	197	1 %

Kaip matyti iš duomenų, pateikiamų lentelėje, sudaryti USB-UART moduliai užima nedaug LPLM lusto resursų. Tokius modulius patogu naudoti sudėtingose sistemose, kai aktualūs lusto resursai. Akivaizdu, kad moduliui su FIFO buferiais reikia daugiau resursų. Atminties elementai, tokie kaip buferiai ar registrai yra sudėtingi ir reikalauja daug aparatinių resursų. Dėl to vykdant sudėtingesnius projektus reikia išnaudoti LPLM luste integruotus specialius registrus ar atminties modulius.

Kadangi buvo siekiama sukurti kuo paprastesnį komunikacijos modulį ir pasiekti kuo didesnę duomenų perdavimo spartą, optimaliausias variantas būtų duomenų perdavimas 1500000 bodų/s greičiu naudojant USB-UART modulį be FIFO buferio. Jis yra paprasčiau realizuojamas ir naudoja mažiau lusto resursų, tačiau, vertinant tai, kad duomenis, patekusius į LPLM reikia perduoti kitiem lusto moduliams, FIFO buferis yra reikalingas siekiant išvengti duomenų praradimo.

USB-UART sąsaja gali būti naudojama duomenų perdavimui tarp kompiuterio ir LPLM lusto. Kriptografiniai skaičiavimai yra viena iš sričių, kurioje būtų galima taikyti didelės spartos UART sąsają. Vienu iš tokių algoritmų galėtų būti WELL512 pseudo atsitiktinių skaičių generatorius. Tai naujos kartos patikimas pseudoatsitiktinių skaičių generatorius [25]. Tokio tipo generatorius gali sugeneruoti 129000000 skaičių per 6,96 s [26]. Generatoriaus vienas skaičius yra 4 baitų ilgio. Nesunku įvertinti, kad tokiam generuojamam duomenų srautui perduoti reikalinga ~68

MB/s duomenų perdavimo sparta. USB-UART tyrimo metu nustatyta, kad galima sėkmingai perduoti duomenis 1,2 Mb/s (150 kB/s) sparta. Tokio greičio nepakaktų generatoriaus kuriamiems srautams perduoti. Net pasiekus stabilų maksimalų galimą greitį 2,4 Mb/s (300 kB/s) generuojamų duomenų nepavyktų perduoti. Šiuo atveju nepakaktų net USB 2.0 duomenų perdavimo spartos – 480 Mb/s (60 MB/s). Tuo tarpu USB 3.0 standartas sugebėtų apdoroti tokius duomenų srautus, tačiau tiek USB 2.0 tiek USB 3.0 protokolai yra sudėtingai realizuojami, todėl WELL512 pseudoatsitiktinių skaičių generatorius yra tinkamas darbui vidinėse sistemose – tik kompiuteryje arba tik FPGA. Duomenimis su kitu įrenginiu šie generatoriai keistis negalėtų.

3.3. Pasiūlymai tobulinimui

Pasiekta 1500000 bodų/s duomenų perdavimo sparta nėra maksimali, kurią leidžia aparatinės įrangos galimybės, sukurtą UART modulį ir komunikacijos programą reikėtų tobulinti. *Nexys3* plokštė turi galimybes keisti taktinį dažnį naudojant PLL ir DCM elementus. Siekiant sumažinti dėl generatoriaus parametrų apvalinimo atsirandančias paklaidas Siekiant sumažinti dėl generatoriaus parametrų apvalinimo atsirandančias paklaidas 3000000 bodų/s duomenų perdavimo spartai galima koreguoti LPLM veikimo dažnį, jį parenkant taip, kad būtų 96 MHz kartotinis. Be to, jeigu didinamas dažnis, galima būtų keisti diskretizacijos daugiklį, taip padidinant ciklų skaičių vienam bitui nuskaityti. Bet to, reikėtų ištirti kaip įtakoja duomenų perdavimo kokybę naudojamų pabaigos bitų skaičius.

Kompiuterio pusėje galimas programinis sprendimas. Naudinga būtų įgyvendinti duomenų patikros protokolą. Įvertinus optimaliausią duomenų paketo dydį galima taikyti ciklinę klaidų patikrą – CRC. Šio metodo esmė – prie duomenų paketo pridedama papildoma tarnybinė informacija apie duomenis. Nusiuntus duomenų paketą į LPLM lustą reikia tikrinti, ar duomenys teisingi. Tam reikalinga įgyvendinti patikros modulį LPLM luste. Patikros modulis kompiuteriui siunčia informaciją apie tai, ar duomenų paketas buvo priimtas sėkmingai. Jeigu atsakymas teigiamas, siunčiamas kitas duomenų paketo dydis. Vertinant galimybes taikyti šį metodą, reikia atkreipti dėmesį į tai, koks yra planuojamas duomenų paketo ilgis ir kokio dydžio papildoma informacija bus siunčiama, kadangi sudarius labai neilgą duomenų paketą nukentės duomenų perdavimo sparta. Vertinant atlikto tyrimo rezultatus, jeigu naudingos informacijos perdavimo sparta dėl nuolat siunčiamos tarnybinės informacijos sumažėja iki 1,2 Mb/s, tuomet taikyti aptartą metodą yra netikslinga.

APIBENDRINIMAS IR IŠVADOS

Magistro baigiamojo darbo tikslas buvo pasiektas išnagrinėjus USB-UART ryšio sąsają ir galimybes ją pritaikyti dideliems duomenų srautams perduoti. XVI-tojoje Lietuvos jaunųjų mokslininkų konferencijoje „Mokslas – Lietuvos Ateitis“ buvo perskaitytas pranešimas „FPGA ir kompiuterio sąsajos tyrimas“ (D priedas). Atlikus ryšio palaikymo sistemos tarp kompiuterio LPLM įgyvendinimą bei tyrimą pasiekti šie rezultatai:

1. Įgyvendinus USB-UART sąsajos modulį buvo pasiekta 3000000 bodų duomenų perdavimo sparta. Naudojant minimą bodinę spartą buvo siunčiami ir priimami duomenys nustatyta, kad galima siųsti duomenis didele sparta, tačiau siunčiant maksimalia įmanoma sparta klaidų skaičius yra per didelis. Kokybiškiausias greitas duomenų perdavimas buvo atliktas su 1500000 bodų per sekundę duomenų perdavimo sparta. Siunčiant duomenis 1500000 bodų/s sparta praradimų ir sugadintų duomenų nebuvo. Todėl galima efektyviai perduoti duomenis 1,2 Mb/s greičiu. Kadangi nebuvo naudojamas joks klaidų taisymo protokolas ir pavyko sėkmingai perduoti duomenis, galima daryti išvadą, kad pakeitus LPLM lusto taktinį dažnį ir pakeitus diskretizacijos dažnį iš 16 į 32 galima tikėtis geresnių rezultatų.

2. Išanalizavus teorinę medžiagą ir atlikus duomenų perdavimo tyrimą buvo suformuoti du pasiūlymai USB-UART moduliams patobulinti keičiant dažnį. Pirmiausia reikėtų įvesti bodinio greičio korekcijas. Skaičiuojant generatoriaus skaitiklio vertę atsiranda paklaidos. Siekiant jas sumažinti galima būtų keisti LPLM lusto taktinį dažnį nuo 100 MHz iki 96 MHz. Toks pakeitimas įtakotų duomenų perdavimą 3000000 bodų per sekundę greičiu, kadangi sinchronizavimo paklaida sumažėtų nuo 4 % iki 0 %.

3. Kitas metodas, padėsiantis pasiekti gerą duomenų perdavimo kokybę yra klaidų taisymo protokolo naudojimas. Kadangi fizinis prievadas turi tik 2 išvadas, aparatinio klaidų taisymo protokolo realizuoti negalima. Šiuo atveju siūlomas programinis klaidų taisymo protokolas. Duomenys siunčiami tam tikro ilgio intervalais, kurių pabaigoje skaičiuojama kontrolinė suma. LPLM lustas, gavęs duomenų paketą tikrina, ar nėra klaidų. Jei klaidų nėra, siunčiamas patvirtinimo signalas, ir laukiama naujų duomenų. Jeigu klaidų rasta, duomenų paketo siuntimas kartojamas. Tokio protokolo taikymas padidintų USB-UART sąsajos patikimumą.

4. Pasinaudojus atlikto tyrimo rezultatais ir nustačius kokybiško ryšio tarp kompiuterio ir periferinio įrenginio svarbą buvo įvertinta galimybė WELL512 pseudo-atsitiktinių skaičių generatoriui siųsti duomenis iš kompiuterio į FPGA. Padaryta išvada, kad 1,2 Mb/s duomenų spartos nepakanka. USB-UART sąsaja būtų nepakankamai sparti net pasiekus maksimalų 2,4 Mb/s duomenų perdavimo greitį.

ŠALTINIAI IR LITERATŪRA

1. Hauck S., DeHorn A. 2008. *Reconfigurable computing: the theory and practice of FPGA-based computation*. United States: Morgan Kaufman Publishers. 945 p.
2. *25 Microchips That Shook The World* [interaktyvus]. 2009. IEEE Spectrum [žiūrėta 2013 m. birželio 3 d.]. Prieiga per internetą: <http://spectrum.ieee.org/semiconductors/processors/25-microchips-that-shook-the-world>
3. *Microprocessors or FPGAs?: Making the Right Choice*. 2011. RTC Magazine [žiūrėta 2013 birželio 3 d.]. Prieiga per internetą: <http://rtcmagazine.com/articles/view/102015>
4. Good T., Benaissa M. 2006. Very small FPGA application-specific instruction processor for AES, *IEEE Transactions on Circuit and Systems I* 53(7), 1447-1486.
5. van Liempd, B. Herrera, D., Figueroa M. 2010. *2010 13th Euromicro Conference on Digital System Design: Architectures, Methods and Tools (DSD)*: 771-778.
6. Aberman, S. 2008. *High-k/Metal-gate Devices for Future CMOS Technology*. Saarbrücken: VDM Verlag. 180 p.
7. *Xilinx Stacked Silicon Interconnect Technology Delivers Breakthrough FPGA Capacity, Bandwidth, and Power Efficiency*. 2012. Xilinx [žiūrėta 2013 birželio 3 d.]. Prieiga per internetą: http://www.xilinx.com/support/documentation/white_papers/wp380_Stacked_Silicon_Interconnect_Technology.pdf
8. *Lowering Power at 28 nm with Xilinx 7 Series FPGAs*. 2012. Xilinx [žiūrėta 2013 birželio 3 d.]. Prieiga per internetą: http://www.xilinx.com/support/documentation/white_papers/wp389_Lowering_Power_at_28nm.pdf
9. *The Death of the Structured ASIC*. 2006. Chip Design Magazine Xilinx [žiūrėta 2013 birželio 3 d.]. Prieiga per internetą: <http://chipdesignmag.com/print.php?articleId=434?issueId=16>
10. *Spartan-6 FPGA Configurable Logic Block*. 2010. Xilinx [žiūrėta 2013 birželio 3 d.]. Prieiga per internetą: http://www.xilinx.com/support/documentation/user_guides/ug384.pdf
11. *Look Out for Lookup Tables*. 2012. All Programmable Planet [žiūrėta 2013 birželio 3 d.]. Prieiga per internetą: http://www.programmableplanet.com/author.asp?section_id=1925&doc_id=244824
12. Perry D. L. 2002. *VHDL: Programming by Example*. USA: McGraw-Hill. 294 p.
13. *The Original USB 2.0 Specification*. 2000. USB Implementers Forum [žiūrėta 2013 birželio 3 d.]. Prieiga per internetą: <http://www.usb.org/developers/docs/>

14. *The USB 3.0 Specification*. 2011. USB Implementers Forum [žiūrėta 2013 birželio 3 d.]. Prieiga per internetą: <http://www.usb.org/developers/docs/>
15. *FT232R USB UART IC*. 2010. FTDI Chip [žiūrėta 2013 birželio 3 d.]. Prieiga per internetą: http://www.ftdichip.com/Support/Documents/DataSheets/ICs/DS_FT232R.pdf
16. Vijay V. Valupadasu R., RamaRao Chunduri B., Kranthi Rekha, Ch., Sreedevi B. 2011. FPGA Implementation of RS232 to Universal Serial Bus Converter, *2011 IEE Symposium on Computers & Informatics*: 237-242.
17. Norhuzaimin J., Maimun H. H. 2005. The Design of High Speed UART, *Asia-Pacific Conference on Applied Electromagnetics*.
18. Roth, Ch. H. 1998. *Digital System Design by using VHDL*. USA: PWS Publishing Company. 534 p.
19. *Nexys3 Board Reference Manual*. 2013. Digilent [žiūrėta 2013 birželio 3 d.]. Prieiga per internetą: http://www.digilentinc.com/Data/Products/NEXYS3/Nexys3_rm.pdf
20. *Nexys3 Board Schematics*. 2011. Digilent [žiūrėta 2013 birželio 3 d.]. Prieiga per internetą: www.digilentinc.com/Data/Products/NEXYS3/NEXYS3_sch.pdf
21. Chu P.P. 2008. *FPGA Prototyping by VHDL Examples: Xilinx Spartan-3 Version*. New Jersey: John Wiley & Sons. 457 p.
22. *Configuring FT232R, FT2232 and FT232B Baud Rates*. 2006. FTDI Chip [žiūrėta 2013 birželio 3 d.]. Prieiga per internetą: http://www.ftdichip.com/Support/Documents/AppNotes/AN232B-05_BaudRates.pdf
23. *FIFO 8x8*. 2010. VHDL and Verilog Designer [žiūrėta 2013 birželio 3 d.]. Prieiga per internetą: <http://vhldesign.blogspot.com/2010/11/fifo-8x8-all-bugs-fixed.html>
24. *Binary file compare with UltraCompare Professional*. 2013. IDM Computer Solutions [žiūrėta 2013 birželio 3 d.]. Prieiga per internetą: <http://www.ultraedit.com/products/ultracompare/tour/binary-file-compare.html>
25. Li Y., Chow P., Jiang J., Zhang M. 2011. Software/Hardware Framework for Generating Parallel Long-Period Random Numbers Using the WELL Method, *21st International Conference on Field Programmable Logic and Applications*: 110-115.
26. Miškinis M. 2013-03-15. Pseudo-atsitiktinių skaičių generatoriaus realizacijos FPGA tyrimas, *Pranešimas 16-ojoje konferencijoje "Mokslas-Lietuvos ateitis"*.

PRIEDAI

A Priedas.

UART komponento be FIFO buferių testavimo programos VHDL kodas

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.numeric_std.all;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity UART is
    generic(
        N: integer := 9;
        M: integer := 2;
        DBIT: integer := 8;
        SB_TICK: integer := 16
    );

    port(
        clk: in std_logic;
        reset: in std_logic;
        rx: in std_logic;
        tx: out std_logic
    );

end UART;

architecture Behavioral of UART is

    type mainState is
        (stReceive,
         stSend);

    signal tick: std_logic;
    --signal tick: std_logic;
    signal rx_done: std_logic;
    signal dout: std_logic_vector(7 downto 0);
    signal tx_start: std_logic;
    signal din: std_logic_vector(7 downto 0);
    signal tx_done: std_logic;
    signal stCur : mainState := stReceive;
    signal stNext: mainState;

    component uart_transmitter is
```

```

generic(
    DBIT: integer := 8;
    SB_TICK: integer := 16
);

port(
    clk, reset: in std_logic;
    tx_start: in std_logic;
    s_tick: in std_logic;
    din: in std_logic_vector(7 downto 0);
    tx_done_tick: out std_logic;
    tx: out std_logic
);
end component;

component uart_receiver is
generic(
    --How many databits are used
    DBIT: integer := 8;
    --How many stop bits are used
    SB_TICK: integer := 16
);
Port ( clk : in  STD_LOGIC;
      reset : in  STD_LOGIC;
      rx : in  STD_LOGIC;
      s_tick : in  STD_LOGIC;
      rx_done_tick : out  STD_LOGIC;
      dout : out  STD_LOGIC_VECTOR (7 downto 0));
end component;

component Counter is
generic(
    N: integer := 5; -- number of bits
    M: integer := 10 -- mod-M
);

port (
    clk, reset: in std_logic;
    max_tick: out std_logic;
    q: out std_logic_vector(N-1 downto 0)
);

end component;

begin

    process (CLK, reset)
    begin
        if (CLK = '1' and CLK'Event) then
            if reset = '1' then
                stCur <= stReceive;
            else
                stCur <= stNext;
            end if;
        end if;
    end if;

```

```

        end process;

UART_RX: uart_receiver
generic map(DBIT, SB_TICK)
port map(clk, reset, rx, tick, rx_done, dout);

UART_TX: uart_transmitter
generic map(DBIT, SB_TICK)
port map(clk, reset, tx_start, tick, din, tx_done, tx);

Tick_counter: Counter
generic map(N, M)
port map(clk, reset, tick, open);

process(stCur, rx_done)
begin
    case stCur is
        when stReceive =>
            tx_start <= '0';
            if rx_done = '1' then
                stNext <= stSend;
            else
                stNext <= stReceive;
            end if;

        when stSend =>
            tx_start <= '1';
            din <= dout;
            stNext <= stReceive;

    end case;
end process;

end Behavioral;

```

B priedas.

UART komponento su FIFO buferiais testavimo VHDL programinis kodas

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

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
use IEEE.NUMERIC_STD.ALL;

use WORK.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity uart_test is

    port(
        clk, reset: in std_logic;
        btn: std_logic_vector(2 downto 0);
        rx: in std_logic;
        tx: out std_logic;
        led: out std_logic_vector(7 downto 0);
        seg: out std_logic_vector(7 downto 0);
        an: out std_logic_vector(3 downto 0)
    );

end uart_test;

architecture Behavioral of uart_test is

    type mainState is
        (stWait, stSend);

    signal tx_full, rx_empty: std_logic;
    signal rec_data, rec_data1 : std_logic_vector(7 downto 0);
    signal btn_tick: std_logic;
    signal wr_uart, rd_uart: std_logic;

    signal stCur : mainState := stWait;
    signal stNext: mainState;

begin

    process (CLK, reset)
    begin
        if (CLK = '1' and CLK'Event) then
            if reset = '1' then
                stCur <= stWait;
            else
                stCur <= stNext;
            end if;
        end if;
    end process;

end architecture;
```

```

    uart_unit: entity work.UART port map(clk, reset, rd_uart, wr_uart,
rx, rec_data1, tx_full, rx_empty, rec_data, tx);
    btn_db_unit: entity work.Debounce port map(clk, reset, btn(0),
open, btn_tick);

    led <= rec_data;
    an <= "1110";

    seg <= '1' & (not tx_full) & "11" & (not rx_empty) & "111";

    process(stCur, rx_empty)
begin
    case stCur is
        when stWait =>
            rd_uart <= '0';
            wr_uart <= '0';
            if rx_empty ='0' then
                stNext <= stSend;
            else
                stNext <= stWait;
            end if;

        when stSend =>
            rd_uart <= '1';
            wr_uart <= '1';
            rec_data1 <= rec_data;
            stNext <= stWait;

    end case;
end process;

end Behavioral;

```

C priedas.

Terminalo programos programos pagrindinis kodas (C#)

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.IO.Ports;
using System.IO;
using System.Diagnostics;

using FTD2XX_NET;

namespace SerialPort
{
    public partial class Form1 : Form
    {

        public Form1()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            if (serialPort1.IsOpen)
            { serialPort1.Close(); }
            StreamWriter sw1 = new StreamWriter(@"C:\Users\Lenovo\Desktop\Kodas\Write.txt");
            StreamWriter sw2 = new StreamWriter(@"C:\Users\Lenovo\Desktop\Kodas\Read.txt");
            Stopwatch TimeCounter = new Stopwatch();
            int size = 10240;
            var chars = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789";
            var random = new Random();
            var result = new string(
                Enumerable.Repeat(chars, size)
                    .Select(s => s[random.Next(s.Length)])
                    .ToArray());

            //Configuring the serial port
            serialPort1.PortName = "COM1";
            serialPort1.BaudRate = 3000000;
            serialPort1.Parity = Parity.None;
            serialPort1.StopBits = StopBits.One;
            serialPort1.Handshake = Handshake.None;
            // serialPort1.ReadTimeout = 500;

            //Constants and arrays

            char[] readArray = new char[size];
            char[] writeArray = new char[size];

            serialPort1.Close();
            serialPort1.Open();
        }
    }
}
```

```

richTextBox1.AppendText("open");

byte[] read_buffer=new byte[size];

TimeCounter.Start();

serialPort1.Write(result);
serialPort1.Read(read_buffer, 0, size);
TimeCounter.Stop();

richTextBox1.AppendText("\n" +
convert.ToString(TimeCounter.ElapsedMilliseconds));

sw2.Write(Encoding.ASCII.GetChars(read_buffer));
sw1.Write(result);
sw1.Close();
sw2.Close();
    }
}
}

```

D priedas.

Dalyvavimą konferencijoje „Mokslas – Lietuvos ateitis“ patvirtinanti pažyma