KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

PROGRAMŲ INŽINERIJOS KATEDRA

Mindaugas Barys

FAILINIŲ SISTEMŲ REALIZACIJŲ, SKIRTŲ MIKROVALDIKLIAMS, TYRIMAS IR TOBULINIMAS

Magistro baigiamasis darbas

Darbo vadovas

Prof. Vacius Jusas

KAUNAS, 2011

KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

PROGRAMŲ INŽINERIJOS KATEDRA

FAILINIŲ SISTEMŲ REALIZACIJŲ, SKIRTŲ MIKROVALDIKLIAMS, TYRIMAS IR TOBULINIMAS

Magistro baigiamasis darbas

Vadovas

Prof. Vacius Jusas

2011 05 27

Recenzentas

Dokt. Tomas Neverdauskas

2011 05 27

Atliko

IFM-9/5 gr. stud.

M. Barys

2011 05 27

mindaugas.barys@gmail.com

KAUNAS, 2011

SANTRAUKA

Elektronika per paskutinius 60 metų labai išsivystė. Šiuolaikiniai mikrovaldikliai yra sudėtingi ir produktyvūs elektronikos įrenginiai. Elektroninės technikos gamintojai stengiasi suteikti savo produkcijai kuo daugiau funkcijų, kad vartotojams ji būtų patogi, kokybiška ir neužimtų daug vietos. Didelė duomenų saugojimo talpa nėra vienintelis privalumas, svarbus ir duomenų įrašymo ir skaitymo greitis. Duomenų perdavimo greitį sąlygoja failinės sistemos pasirinkimas pagal poreikius. Todėl failinių sistemų tyrimas ir tobulinimas labai aktualus, dėl esamos paklausos rinkoje. Šiuo metu yra realizuota nemažai failinių sistemų mikrovaldikliams (FAT16, FAT32 ir kt.), tačiau iš jų gausos sunku išskirti ir pasirinkti sparčiausią, bei kitus vartotojo keliamus reikalavimus atitinkančią sistemą. Todėl labai svarbu aptarti kokia failinių sistemų elgsena, kokie privalumai pasireiškia skirtingose realizacijose ir ištirti jų duomenų rašymo bei skaitymo greičius. **Tyrimo tikslas** - ištirti failinę sistemą skirtą mikrovaldikliui ir ją patobulinti.

Tyrimo uždaviniai:

1) atskleisti failinių sistemų realizacijų skirtų mikrovaldikliams teorinius aspektus;
2) ištirti failinių sistemų realizacijų skirtų mikrovaldikliams savybes ir funkcijas;
3) patobulinti failinės sistemos realizacijos skirtos mikrovaldikliui duomenų perdavimo greitį;

Tyrimas atskleidė, kad taikant FPGA programuojamąją matricą didinant taktinį programinio mikrovaldiklio dažnį, duomenų perdavimo greitis proporcingai didėja.

SUMMARY

In the last 60 years electronics has developed significantly. Modern microcontrollers are complex and efficient electronic devices. Electronic equipment manufacturers are trying to make their products with as many features as possible, that it would be convenient for consumers, had a high quality and don't occupy too much space. Large data storage capacity is not the only advantage, the data write and read speed also matters. Data transfer rate is determined in accordance with choice of filesystem needs. Therefore filesystem research and development is very relevant to the current market demand. At present there are implemented a number of filesystem for microcontroller (FAT16, FAT32 and ect.), but their majority make it difficult to choose the fastest and other requirements of the user corresponding system. Therefore, it is important to discuss filesystem behavior, what advantages occurs in different implementations and analyze their data recording and reading speeds.

The purpose of the research: examine the filesystem for the microcontroller and improve it.

Objectives:

1) to reveal filesystem implementations for the microcontroller theoretical aspects;
2) examine filesystem implementations for the microcontroller features and functions;
3) improve filesystem implementations for the microcontroller data transfer rate;

The investigation revealed that using FPGA to increase microcontroller frequency increases the data transfer rate.

SVARBIŲ TERMINŲ ŽODYNĖLIS

FAT – failinė sistema arba failų išdėstymų lentelė.

Klasteris - vienetas, sudarytas iš kelių vieno tipo elementų, kuris veikia kaip vieninga sistema, turinti tam tikrų savybių.

SD kortelė **-** tai kortelės formos skaitmeninių duomenų laikmena, naudojama duomenims kaupti.

USB - universalioji jungtis, naudojama kompiuteriuose.

SDHC - padidintos talpos SD kortelė.

SPI - duomenų perdavimo sąsaja naudojama atminties kortelėse.

FPGA - viena iš įterptinių sistemų rūšių,suteikianti galimybė patikrinti lustą prieš paleidžiant jį į gamybą.

HDL - aparatūros aprašymo kalbos.

VHDL - viena iš aparatūros aprašymo kalbų.

Verilog - viena iš aparatūros aprašymo kalbų.

RTL - abstrakcijos lygis naudojamas aprašant sinchroninės skaitmeninės grandinės veikimą.

RAM - operatyvioji atmintis.

PIC - General Instrument kompanijos sukurtas mikrovaldiklis.

AVR - Atmel kompanijos sukurtas mikrovaldiklis.

LENTELĖS

PAVEIKSLAI

PRIEDAI

TURINYS

ĮŽANGA

Šiais technologijų laikais elektronikos prietaisai (kompiuteriai, mobilieji telefonai, skaitmeniniai grotuvai ir t.t.) tampa vis sudėtingesni ir mažesni, juose saugomų duomenų kiekis sparčiai didėja. Kiekviename iš minėtų prietaisų naudojama informacija, kuri saugoma elektroninėse laikmenose, tačiau be failinės sistemos saugiai kaupti informacijos laikmenoje neįmanoma. Mikrovaldikliai skirti susieti vartotojo sąsaja su failine sistema prietaisuose bei visur kur naudojama skaitmeninė elektronika. Elektronika per paskutinius 60 metų labai išsivystė. Šiuolaikiniai mikrovaldikliai yra sudėtingi ir produktyvūs elektronikos įrenginiai. Juos galima sutikti automobiliuose, buitinėje įrangoje, pramonėje. Šiuo metu mikrovaldiklius gamina praktiškai visi integrinių grandynų gamintojai. Elektroninės technikos gamintojai stengiasi suteikti savo produkcijai kuo daugiau funkcijų, kad vartotojams ji būtų patogi, kokybiška ir neužimtų daug vietos, todėl projektuotojams iškyla problema, kaip patenkinti vartotojų poreikius ir suteikti jiems galimybę naudotis technikos teikiamomis funkcijomis ir talpinti joje norimą duomenų kiekį. Kaip kompromisas projektuotojams – didelės talpos elektroninės laikmenos (SD kortelės ar standieji diskai), kurios kuriamos vis mažesnės. Didelė duomenų saugojimo talpa nėra vienintelis privalumas, svarbus ir duomenų įrašymo ir skaitymo greitis. Duomenų perdavimo greitį sąlygoja failinės sistemos pasirinkimas pagal poreikius. Todėl failinių sistemų tyrimas ir tobulinimas labai aktualus, dėl esamos paklausos rinkoje. Baigiamajame magistro darbe nagrinėjama ši aktuali tematika, atliekamas tyrimas ir nuspėjamos tobulino galimybės.

Failinės sistemos, skirtos mikrovaldikliams, pasirinkimas asmeninis vartotojo reikalas, tačiau dažnu atveju pasirinkimą lemią vienos sistemos pranašumas prieš kitas. Norint išsiaiškinti failinių sistemų realizacijų pranašumus ir trūkumus būtina atlikti išsamų tyrimą, bei kruopščiai jas išanalizuoti.

Šiuo metu yra realizuota nemažai failinių sistemų mikrovaldikliams (FAT16, FAT32 ir kt.), tačiau iš jų gausos sunku išskirti ir pasirinkti sparčiausią, bei kitus vartotojo keliamus reikalavimus atitinkančią sistemą. Todėl labai svarbu aptarti kokia failinių sistemų elgsena ir kokie privalumai pasireiškia skirtingose realizacijose ir ištirti jų duomenų rašymo ir skaitymo greičius. Atsakymų paieška į šiuos klausimus sudaro tyrimo **problemos** pagrindą.

Tyrimo tikslas - ištirti failinę sistemą skirtą mikrovaldikliui ir ją patobulinti.

Tyrimo uždaviniai:

4) atskleisti failinių sistemų realizacijų skirtų mikrovaldikliams teorinius aspektus;
5) ištirti failinių sistemų realizacijų skirtų mikrovaldikliams savybes ir funkcijas;
6) patobulinti failinės sistemos realizacijos skirtos mikrovaldikliui duomenų perdavimo greitį;

1. FAILINIŲ SISTEMŲ REALIZACIJŲ SKIRTŲ MIKROVALDIKLIAMS TEORINIAI ASPEKTAI

Failų sistema yra pagrindinė struktūra, kurią naudoja kompiuteris tvarkyti duomenis ir standųjį diską. Jeigu diegiamas naujas standusis diskas ar kita elektroninė laikmena, būtina juos suskirstyti ir formatuoti naudojant failų sistemą prieš pradedant saugoti duomenis ir programas. Šiuo metu populiariausios failų sistemos, yra: NTFS, FAT16, FAT32 (Windows operacinėje sistemoje) ir EXT giminės sistemos.

1.1. FAT failinė sistema

Failų išdėstymo lentelė FAT (angl. *File Allocation Table*) - failinė sistema sukurta 1970m. Maždaug po dešimties metų 1980m. Ši failinė sistema buvo pritaikyta ir pradėta naudota Microsoft kompanijos MS-DOS operacinėje sistemoje. Iš pradžių sistema buvo gana paprasta ir pritaikyta diskeliams (angl. *floppy*) su mažesne nei 500 KB dydžio talpa. Bėgant laikui programuotojai ir inžinieriai tobulino naująją failinę sistemą, kad ji būtų pritaikyta didesnėms elektroninėms laikmenoms. Šiuo metu yra keturios FAT failinės sistemos: FAT12, FAT16, FAT32 ir exFAT kartais vadinama FAT64. Pagrindinis skirtumas tarp šių sistemų – pavadinimai, kurie kilę nuo bitų kiekio, skirtų vienam FAT įrašui [10].

1 lentelė. FAT šeimos failinių sistemų pagrindiniai duomenys

FAILINĖS SISTEMOS PAVADINIMAS	MAKSIMALUS KLASTERIŲ SKAIČIUS	KLASTERIŲ DYDIS	ELEKTRONINĖS LAIKMENOS MAKSIMALI TALPA
FAT12	4086	Nuo 0,5 KB iki 4 KB	32 MB
FAT16	65526	Nuo 2 KB iki 32 KB	2 GB
FAT32	268435456	Nuo 4 KB iki 32 KB	8 TB
exFAT	2^{64}	Iki 32 MB	16 EB

Ši failų sistema atsirado kartu su asmeniniais kompiuteriais ir buvo skirta failus saugoti diskuose. Informacija saugoma skirsniais po 512 baitų. Visas diskas suskirstytas į fiksuoto ilgio sritis vadinamuosius klasterius (angl. *cluster*). Failų išdėstymo lentelė (FAT) – tai skaitmeninių reikšmių sąrašas, kuriame aprašomas klasterių pasiskirstymas standžiojo disko skirsnyje t.y. kiekvieno klasterio būsena tame skirsnyje, kuriame jis randasi. Iš tiesų kiekviena pasiskirstymo lentelės ląstelė (angl. *cell*) atitinka klasterį. Kiekviena ląstelė turi numerį (skaičių rinkinį), kuris parodo ar klasteris yra užimtas failo, jei taip, nurodoma sekančio klasterio vieta faile. Tokiu būdu susiduriama su FAT grandine, susidedančią iš tarpusavyje susijusių nuorodų, rodančių sekančius klasterius iki failo pabaigos. Kiekvienas FAT įrašas yra 12,16 arba 32 bitų ilgio (priklausomai nuo pasirinktos FAT sistemos). Du pirmieji įrašai kaupia informaciją apie lentelę, kai tuo tarpu sekantys nurodo klasterius. Tam tikri įrašai gali būti sudaryti iš reikšmių nurodančių specifinio klasterio būseną. Pavyzdžiui reikšmė *0000* nurodo, kad klasteris nenaudojamas, *FFF7* pažymi

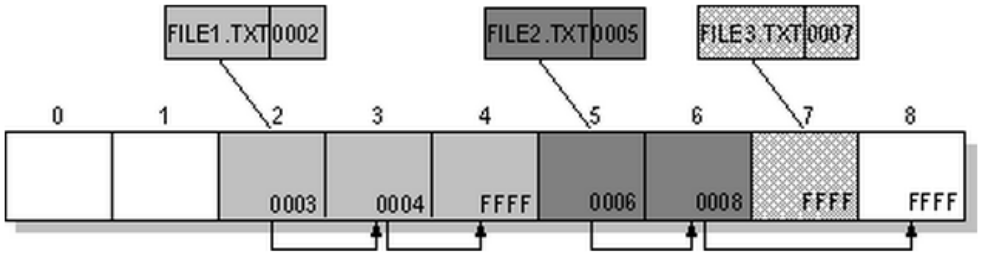
blogą klasterį, kad jis nebūtų naudojamas ir reikšmės tarp FFF8 ir FFFF nurodo, kad klasteryje yra failo pabaiga. Kiekvienas skirsnis (angl. *partition*) turi dvi lentelių kopijas, saugomas greta viena kitos diske, todėl yra galimybė atkurti duomenis jei pirma lentelės kopija bus sugadinta.

Šakninis aplankas turi įrašą apie kiekviena failą ir aplanką. Vienintelis skirtumas tarp šakninio ir kitų aplankų - šakninis aplankas randasi specifinėje vietoje diske ir yra fiksuoto dydžio (512 įrašų standžiajame diske).

Aplankams skirti 32 baitai įrašams kiekvienam failui ir aplankui esančiam viduje. Įrašas apima šią informaciją:

- Vardas;
- Atributo baitas (8 bitai)
- Sukūrimo laikas (24 bitų)
- Sukūrimo data (16 bitų)
- Paskutinio modifikavimo laikas (16 bitų)
- Paskutinio modifikavimo data (16 bitų)
- Pirminio klasterio numeris FAT sistemoje (16 bitų)
- Failo dydis (32 bitai)

Nėra jokios organizacijos FAT aplanko struktūroje ir failai keliami į pirmą prieinamą vietą disko talpoje. Pirminis klasterio skaičius yra failo naudojamo pirmo klasterio adresas. Kiekvienas klasteris turi rodyklę į kitą klasterį faile, arba nurodo kad šis klasteris randasi failo pabaigoje. Visa tai atvaizduojama žemiau pateiktame paveikslėlyje (žr. 1 pav.).



1 pav. **Failų pasiskirstymo lentelė [12]**

Paveikslėlyje pateikiami trys failai. *File1.txt* failas yra pakankamai didelis, kad naudoti tris klasterius. Mažesnis failas *file3.txt* telpa viename klasteryje. Trečiasis failas *file2.txt* yra suskaldytas failas ir taip pat naudoja tris klasterius. Kiekvienu atveju nurodomas failo naudojamo pirmojo klasterio adresas.

1.1.1. FAT12

Pradinė FAT failinės sistemos versija dabar yra vadinama FAT12. Ji buvo kuriama ir naudojama diskeliuose. Vienam FAT12 failinės sistemos FAT įrašui skirta 12 bitų, limituotas klasterių skaičius iki 2^{12} = 4096 (realus klasterių skaičius yra mažesnis, nei skaičiuojant

teoriškai, nes dalis jų laikomi rezervuotais pačiai failų sistemai) ir elektroninės laikmenos talpa negali siekti daugiau nei 32 MB [5]. Daugiau nei prieš 30 metų FAT12 buvo naudojama kelių informacinių technologijų gamintojų įvairiuose fiziniuose formatuose, tačiau dažniausiai tipiškuose to laikmečio diskeliuose, kurie buvo vienpusiai, 5,25 colio skersmens, turėjo 40 koncentrinių takelių (viename takelyje buvo 8 sektoriai), bendra talpa siekė 160 KB. FAT12 failinės sistemos limitai viršijo šią talpą daugiau nei 10 kartų. Šiuo metu FAT12 galime aptikti senesnio tipo, bei naujesniuose 1,44 MB talpos diskeliuose.

1.1.2. FAT16

1984 kompanija IBM išleido į prekybą naują kompiuterį pavadintą PC AT, kurio standžiojo disko talpa siekė 20 MB. Beveik tuo pačiu metu Microsoft pristatė MS-DOS 3.0 operacinę sistemą. Tai lėmė naujos failinės sistemos FAT16 atsiradimą. Vienam FAT16 failinės sistemos FAT įrašui buvo skirta 16 bitų, tai leido padidinti klasterių skaičių iki 65536, bet standžiojo disko skirsnio (angl. *partition*) talpa nepasikeitė ir išliko 32 MB, kaip ir ankstesnėje FAT12 sistemoje. Tačiau ši failinė sistema nebuvo tokia FAT16 kokią žinome šiandien, nes 1988 metais po MS-DOS 4.0 sukūrimo ji buvo patobulinta. Klasterių dydis išaugo iki 32 KB, dėl šios priežasties žymiai šoktelėjo standžiojo disko skirsnio dydis. Maksimalus standžiojo disko skirsnio dydis apskaičiuojamas klasterių skaičių padauginus iš klasterio dydžio, taigi atlikus šią nesudėtinga matematinę operaciją gaunamas neįtikėtinas to laikmečio disko skirsnio dydis – 2 GB [11].

Būtina atkreipti dėmesį į tai, jog failas gali užimti tik sudėtinį klasterių skaičių, tai reiškia, kad failui užimant kelis klasterius, paskutinis eilėje bus užimtas tik dalinai ir neužimta vieta liks nenaudojama. Todėl kuo mažesnis klasteris yra naudojamas, tuo mažiau iššvaistoma vietos. Apskaičiuota, kad failas išeikvoja vidutiniškai puse klasterio, tai reiškia, jog naudojant 2 GB disko skirsnį – 16 KB iššvaistoma vienam failui.

1.1.3. FAT32

Tam, kad pralenkti FAT16 talpos limitą, Microsoft dar kartą išplėtė klasterio dydi ir naują kūrinį pavadino FAT32. Klasterio reikšmės atitinka 32 bitų skaičius, iš kurių 28 bitai skirti laikyti klasterių skaičių, maksimaliai gali būti 268 milijonų (2^{28}) klasterių, likę 4 bitai yra rezervuoti. Disko skirsnio dydis teoriškai išaugo iki 8 TB, tačiau kūrėjai savavališkai apribojo iki 32 GB limito Windows 9x sistemoms, kad padidinti NTFS failinės sistemos naudojimą ateityje. Kadangi FAT32 failinė sistema gali būti sudarytą iš daugiau klasterių, nei FAT16 įmanoma žymiai sumažinti klasterių dydį ir tuo pačiu sutaupyti švaistomą disko vietą. Pavyzdžiui, naudojant 2 GB skirsnį įmanoma naudoti 4 KB klasterius su FAT32 sistema (vietoje 32 KB klasterių su FAT16), dėl to disko vietos švaistymas sumažėjo 8 kartus.

1.2. Failinių sistemų taikymo aplinkos

Dvi pagrindinės failinių sistemų taikymo aplinkos yra standieji diskai ir mobilios elektroninės laikmenos (atminties kortelės, USB raktai ir pan.)

Standusis diskas – duomenų saugojimo įrenginys, kuris pasižymi ypač didele duomenų saugojimo talpa ir duomenų perdavimo greičiu. Šis įrenginys ar įrenginių grupė sujungti į viena visumą yra pagrindinė kompiuterio duomenų talpykla.

Atminties kortelė, kartais vadinama atmintuke (angl. *flash memory card, storage card*) - tai kortelės formos skaitmeninių duomenų laikmena. Naudojama duomenims kaupti bei perkelti mobiliuose telefonuose, muzikos ir vaizdo grotuvuose, kompiuteriuose ir kituose prietaisuose. Labiausiai paplitusios yra SD ir microSD kortelės.

1.2.1. SD kortelė

Per pastaruosius 12 metų, mobiliųjų telefonų, skaitmeninių vaizdo kamerų, mp3 ir vaizdo grotuvų pardavimai žymiai išaugo. Tuo pačiu didėjo patogaus duomenų saugojimo būdo poreikis vartotojų naudojamuose mobiliuose įtaisuose. Atminties kortelė rinkoje buvo pristatyta kaip produktas visiškai atitinkantis vartotojų poreikius - maža duomenų saugojimo laikmena, pasižyminti didele atminties talpa.

1997m. buvo sukurtos ir pateiktos rinkoje MMC (angl. *Multi Media Card*) kortelės. Jų sėkmės priežastis - mažas dydis ir didelis duomenų tankis, kurio buvo pasiekta naudojant NAND-Flash technologiją. Tačiau 1999m. elektronikos įmonės Panasonic, SanDisk ir Toshiba susijungė bendram projektui ir sukūrė SD (angl. *Secure Digital*) kortelių standartą, kuris yra MMC kortelių patobulinimas, todėl dažniausiai įrenginiai gali naudoti ir MMC ir SD tipo korteles. Dėl šių atnaujinimų duomenų perdavimo magistralės plotis padidėjo iki 4 bitų, o fizinis kortelės plotis tapo dvigubai storesnis. Atminties kortelės greitis buvo žymiai padidintas, o talpa išaugo iki 2Gb.

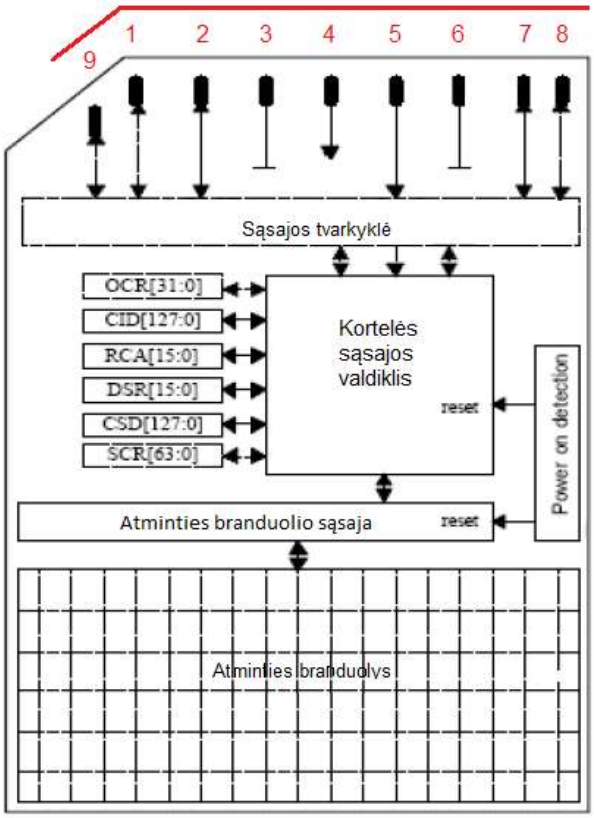
SD kortelės dydis yra 32mm ilgis, 24mm plotis ir 2.1mm storis. Šiomis dienomis SD kortelių talpa svyruoja nuo 8 MB iki 32 GB. Didesnės nei 4 GB SD kortelės yra gaminamos naudojant SDHC (angl. *Secure Digital High Capacity Card*) technologiją. Kadangi SD ir SDHC kortelės nesiskiria nei dydžiu, nei forma, tai gali sukelti tam tikrų nesklandumų vartotojams, naudojant jas senesniuose įrenginiuose, nes nevisi palaiko SDHC formato.

SD kortelės perduoda duomenis 10-20 Mbit/s greičiu, bet šis dydis gali svyruoti, dėl naujų technologijų taikymo ir greičio perdavimo spartinimo naujuose įrenginiuose. Šios kortelės pasižymi funkcija, leidžiančia uždrausti rašymą, kuri įjungiama nuspaudus jungiklį kortelės korpuse. Nuspaudus šį jungiklį, kortelė užrakinama ir leidžiama tik informacijos nuskaitymas- taip išvengiama atsitiktinio duomenų sunaikinimo.

Svarbu aptarti SD kortelės evoliuciją ir išskirti svarbiausius jos tobulinimo laikotarpius:

- 2000m. SD kortelių veikimo dažnio diapazonas buvo 400Khz-25 Mhz, tai teoriškai leido pasiekti 100 Mbit/s duomenų perdavimo greitį ir talpa buvo ribojama iki 2^{31} arba 2 GB.
- 2006m. buvo pristatyta SDHC kortelė veikianti iki 50 Mhz dažniu, teoriškai jos maksimalus greitis – 200 Mbit/s. Buvo pakeistas duomenų perdavimo adresavimas nuo vieno baito į bloką į 512 baitus. Talpa padidėjo iki 32Gb.
- 2009m. pradžioje buvo pristatyta naujos kartos SD kortelės specifikacija: SDXC. Pranešimo spaudai teigiama, kad kartelės greitis sieks 832 Mbit/s, o talpa iki 2 TB. Norint pasiekti šį perdavimo greitį, bus reikalingas 100 Mhz dažnis.

Peržvelgus pateiktus duomenis, galima teigti, kad SD kortelių greitis sistemingai didėja ir iki šių dienų greitis nėra maksimaliai išvystytas, todėl paliekama terpė tolimesniems tyrimams.



2 pav. **SD kortelės architektūra[17]**

Pateiktame paveiksle galima matyti SD kortelės architektūrą (žr. 2 pav.). Kortelė susideda iš dviejų sąsajų: išorinės ir vidinės. Išorinė sąsaja susideda iš 9 komunikavimo kontaktų (angl. *pins*), sukurtų veikti žemos įtampos diapazone, kurie išdėstyti kortelės viršuje. Schemoje vaizduojami kontaktai paaiškinti žemiau pateiktoje lentelėje.

2 lentelė. SD kortelės kontaktų sąrašas

KONTAKTO NR.	PAVADINIMAS	FUNKCIJA (SD MAGISTRALĖ)	FUNKCIJA (SPI MAGISTRALĖ)
1	DAT3/CS	Duomenų linija 3	Chip select/Slave select (SS)
2	CMD/DI	Komandos linija	Master out slave in (MOSI)
3	VSS1	Įžeminimas	Įžeminimas
4	VDD	Maitinimas	Maitinimas
5	CLK	Takto laikrodis	Takto laikrodis (SCK)
6	VSS2	Įžeminimas	Įžeminimas
7	DAT0/DO	Duomenų linija 0	Master in slave out (MISO)
8	DAT1/IRQ	Duomenų linija 1	Nenaudojamas arba IRQ
9	DAT2/NC	Duomenų linija 2	Nenaudojamas

Iš 1 lentelės galima matyti, kad daugelis SD kortelės kontaktų yra dvejopos paskirties (žr. 2 lentelė). SD kortelės palaiko tris duomenų perdavimo protokolus, kurie skirstomi i dvi klases besiskiriančias viena nuo kitos. Lentelėje pateiktos kortelės kontaktų funkcijos SPI ir SD režimuose.

Antra SD kortelės sąsaja yra vidinė sudaryta iš: sąsajos tvarkyklės (angl. *interface driver*), kortelės sąsajos valdiklio (angl. *card interface controller*), registrų rinkinio (žr. 3 lentelė), atminties branduolio sąsajos ir atminties branduolio. Šiuolaikinės kortelės taip pat turi vidinį duomenų buferį. Sąsajos tvarkyklė persiunčia komandinius ir duomenų signalus kortelės sąsajos valdikliui. Kortelės sąsajos valdiklis apdoroja signalus ir atlieka nurodytas užduotis. Užduočių rinkinys susideda iš duomenų skaitymo ir rašymo į registrų, bei duomenų pernešimo ir rašymo į atmintį. Valdiklis taip pat yra atsakingas už klaidų aptikimą ir jų taisymą. Tobulesni modeliai turi integruotus nusidėvėjimo algoritmus, kurie skirti prailginti kortelės tarnavimo laiką.

3 lentelė. SD kortelės vidiniai registrai

PAVADINIMAS	**ILGIS BITAIS**	**APRAŠYMAS**
CID	128	Individualus kortelės identifikacijos numeris (angl. *Card Identification*).
RCA	16	Santykinis kortelės adresas (angl. *Relative Card Address*).
DSR	16	Tvarkyklių būsenų registras (angl. *Driver Stage Register*).
CSD	128	Informacija apie kortelės operacijų būklę (angl. *Card Specific Data*).
SCR	64	Informacija apie SD atminties kortelės specialias funkcijas (angl. *SD Configuration Register*).
OCR	32	Veikimo sąlygų registras (angl. *Operation conditions register*).
SSD	512	Informacija apie kortelės nuosavybės ypatybės (angl. *SD Status*).
CSR	32	Informacija apie kortelės būsena (angl. *Card Status*).

Iš 3 lentelės matoma, kad vidinė kortelės sąsaja sudaryta iš 9 registrų. OCR, CID, CSD ir SCR registrai saugo juose patalpintą kortelės konfigūraciją. RCA registras saugo kortelės

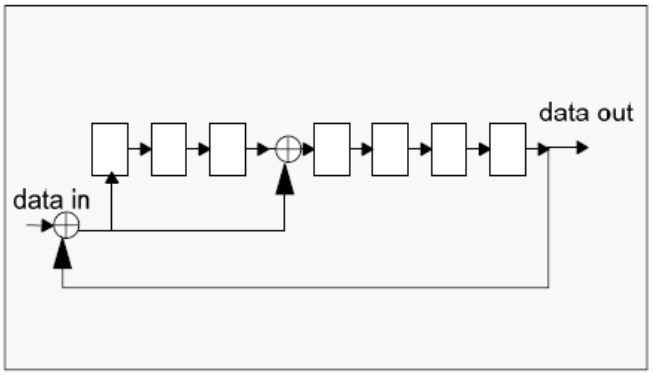
santykinį komunikacijos adresą ir duomenis apie dabartinę sesiją. SSD ir CSR registrai saugo komunikavimo protokolus susijusius su kortelės būkle.

SD kortelių duomenų perdavimo magistralės

SD magistralę sudaro ankščiau paminėti kortelės kontaktai, kurie dalyvauja duomenų perdavimo procese. Šios magistralės duomenų perdavimo protokolas gali būti 1 arba 4 bitų ilgio.

SD 1-bito duomenų perdavimo protokolas yra sinchroninis ir nuoseklus, turintis viena duomenų perdavimo liniją, kuri skirta didelio duomenų kiekio perdavimo, vieną takto laikrodžio (CLK) liniją ir viena valdymo (CMD) liniją, naudojamą komandų siuntimui. Valdymo ir duomenų signalo perdavimas vykdomas, laikant CMD ir DAT0 žemą frontą. SD 1-bito protokolas tiesiogiai palaiko magistralės paskirstymą. Paprasta *slave-master* schema leidžia kelioms SD kortelėms dalintis vienu takto laikrodžiu (CLK) ir DATA0 linija.

SD4-bitų protokolas perdavimo protokolas yra beveik identiškas prieš tai minėtam SD 1-bito protokolui. Pagrindinis skirtumas tarp jų yra magistralės plotis – didesnio kiekio duomenų persiuntimui naudojama 4 bitų magistralė vietoje vienos linijos. Ją sudaro 6 komunikavimo linijos ir 3 energijos tiekimo linijos. Prie DAT ir CMD komunikavimo linijų turi būti prijungti *pull-up* rezistoriai, norint išvengti įvairių svyravimų kai kortelė būna ištraukiama. Naudojant tinkamą dizainą, sukuriamas potencialas didelio duomenų kiekio keturgubam perdavimo pralaidumui. Tiek SD-1, tiek SD-4 bitų protokolai, pagal nutylėjimą reikalauja CRC (angl. *Cyclic Redundancy Check*) didelio kiekio duomenų perdavimui. CRC yra gana paprastas klaidų aptikimo metodas (žr. 3 pav.), skirtas apsaugoti SD kortelės komandas, atsakus, ir duomenų perdavimą nuo siuntimo klaidų galinčių įvykti SD kortelės magistralėje. CRC yra generuojamas kiekvienai komandai ir tikrinamas kiekvienas CMD linijos atsakymas, o duomenų blokai tikrinami kai yra sugeneruoti ir paruosti siųsti.



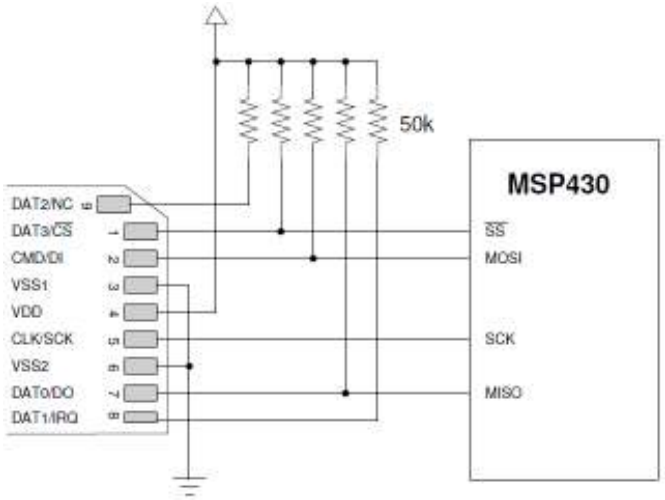
3 pav. **CRC klaidų paieškos generatorius [19]**

SD 4 – bitų režime įeinantys duomenys yra padauginami keturioms magistralės (DAT) linijoms ir 16 – bitų CRC yra nepriklausomai apskaičiuojamas kiekvienai iš keturių linijų. Bet kokios programinės įrangos kūrimui, CRC skaičiavimas tokiomis sąlygomis gali būti toks sudėtingas, kad menkina 4 - bitų magistralės privalumus. 4 – bitų paralelinis CRC nėra toks

svarbus kuriant aparatūrą, tačiau dažnai pasirenkamuose ASIC arba FPGA platesnės magistralės naudojimas išlieka privalumu.

Trečiasis SD kortelės duomenų perdavimo protokolas – SPI (angl. *Serial Peripheral Interface*). SPI - komunikavimo magistralė, sukurta Motorola kompanija. Ši magistralė veikia dvipusiu (angl. *duplex*) režimu, kai duomenų perdavimas gali vykti vienu metu su duomenų priėmimu. Kartais ši magistralė vadinama „keturlaide“ nuosekliąja magistrale, nes SPI magistralė naudoja keturis loginius signalus. Nuo 1 ir 4 – bitų protokolų jis skiriasi tuo, kad veikia per bendras ir gerai žinomas magistralės sąsajas. SPI yra sinchroninis nuoseklus protokolas, kuris itin populiarus išoriniuose mikrovaldiklių įrenginiuose. Dauguma šiuolaikinių mikrovaldiklių, įskaitant ir ATmega328, palaiko ši perdavimo protokolą kaip vieną iš standartinių, dėl gana spartaus duomenų perdavimo. SPI komunikavimo režimas palaiko tik SD kortelės duomenų perdavimo protokolo dalį. Tačiau dauguma nepalaikomų komandų rinkinių yra paprasčiausiai nereikalingi SPI režime. Pilnai funkcionuojanti SD kortelės realizacija gali būti realizuota naudojant SPI režimą.

Šios duomenų perdavimo magistralės populiarumas ir nesudėtingas naudojimas mikrovaldikliuose, suteikia projektuotojui didelį privalumą. Naudojant SPI projektuotojas gali pasirinkti greitas lygiagrečias sąsajas arba priklausomai nuo taikymo, gali rinktis lėtesnes realizacijas. Tarp dviejų pagrindinių duomenų perdavimo režimų aptinkama nežymių skirtumų vykdant inicializacija.



4 pav. **SD kortelės schematinis vaizdas – naudojant SPI režimą [18]**

Paveiksle (žr. 4 pav.) vaizduojama SD kortelės elektroninė sąsaja SPI režimui. Pavaizduoti išoriniai *pull-up rezistoriai* pagal SD protokolą yra reikalingi, net ir nenaudojamuose duomenų kontaktuose. Svarbu paminėti, kad SS kontaktas turi gauti pastovų loginį vienetą, kol SPI linija siunčiamos 8 bitų sekos ir tuo pačiu negali būti valdomas mikrovaldiklio kaip automatinis *slave-selection* kontaktas, o turi būti valdomas programiškai.

1.3. Skaitmeninės elektroninės aparatūros aprašymo kalbos

Skaitmeninė sistema gali būti apibūdinama skirtinguose abstrakcijų lygiuose. Projektavimo procesai keičiasi, nes laikui bėgant tobulėja projektuotojų įgūdžiai ir programinė įranga. Reikalinga bendra, viską apjungianti kalba, norint keistis informacija tarp projektuotojų ir programinės įrangos. Tam skirta - aparatūros aprašymo kalba (anlg. Hardware Description Language). HDL – turi tiksliai modeliuoti ir aprašyti jau sukurtą arba kuriamą grandinę, struktūrinio arba funkcinio aprašymo požiūriu, priklausomai nuo abstrakcijos lygio. Nes ši aprašymo kalba modeliuojama pagal aparatūrą, jų semantika ir naudojimas labai skiriasi nuo tradicinių programavimo kalbų. Šiame skyriuje aptariamos populiariausios aparatūros aprašymo kalbos. VHDL ir Verilog yra dvi dažniausiai naudojamos HDL kalbos. Nors jų sintaksė ir „išvaizda“ labai skirtinga, jų galimybės ir taikymo sritys yra panašios. Šios kalbos yra pramoninis standartas ir yra palaikomos daugelio programinių įrangų. Todėl svarbu išnagrinėti jų pagrindines funkcijas ir pritaikymą, pateikiant kalbos sintaksių pavyzdžius.

Skaitmeninės elektroninės aparatūros aprašymo kalbų taikymas

Norint geriau suprasti HDL, naudinga išnagrinėti šios aparatūros kodo panaudojimą. Tradicinėse programavimo kalbose, kodas dažniausiai rašomas išspręsti specifinei problemai. Paimamos įėjimų išvadų reikšmės ir atitinkamai generuojamos išėjimų išvadų reikšmės. Programos kodas pirmiausia yra kompiliuojamas ir paleidžiamas kompiuteryje. Kita vertus HDL programos kodo taikymas labai skiriasi. Galima išskirti aparatūros aprašymo kalbų tris pagrindines ypatybės[14]:

Formali dokumentacija. Skaitmeninės sistemos įprastai pradedamos kurti nuo dokumentacijos rašymo. Kadangi žmonių kalba nėra tiksli, apibūdinimai neišbaigti ir dviprasmiški ir tas pats apibrėžimas gali būti skirtingai traktuojas ir interpretuojamas. Dėl HDL semantikos ir sintaksės, kuri yra griežtai apibrėžta, sistemos specifikuojamos HDL kalba yra tikslios ir aiškios. Todėl ši kalba gali būti naudojama naudotojų ir projektuotojų, kaip sistemos specifikacijos ir dokumentacijos aprašas.

Įėjimo išvadai modeliavimo programai. Modeliavimas yra naudojamas tirti ir tikrinti grandinės operacijas, nekuriant fizinės sistemos. Modeliavimo programa suteikia pagrindą modeliuoti veikiančias operacijas kompiuteryje ir turi tokią funkciją, kaip kalbos sintaksės konstravimą. HDL kodas, sujungta testu ir duomenų rinkinio failu, generuoja *teshbench*, kuris tampa vienas pagrindiniu failu modeliavimo programai. Modeliavimo metu interpretuojamas HDL kodas ir atitinkamai generuojamas atsakas.

Įėjimo išvadai sintezavimo programai. Modernus vystymasis pagrįstas tobulinimo procesu, kuris palaipsniui keičia aukštesnio lygio funkcinį aprašą į žemesnio lygio struktūrinį. Kai kurie tobulinimo žingsniai gali būti atliekami sintezės programos. Ši programa naudoja HDL kodą kaip įėjimo išvadą ir realizuoja grandinę iš naudojamos bibliotekos komponentų. Sintezatoriaus išėjimo išvadai yra naujas HDL kodas, kuris sudaro sintezuotos grandinės struktūrinį aprašą.

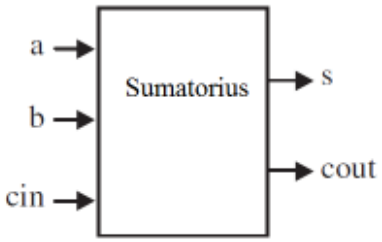
1.3.1. VHDL

VHDL kūrimas buvo remiamas Amerikos saugumo departamento lėšomis, kaip aparatūros aprašymo standartas ankstyvuosiuose 1980m. ir tuomet buvo perleista į IEEE (elektrotechnikos ir elektronikos inžinierių institutas). Tuomet tai tapo standartu šioje srityje.

VHDL – skaitmeninės elektroninės aparatūros funkcionavimo aprašymo kalba. Ši kalba yra viena aparatūros aprašymo kalbų, kurios laikomos standartu. Kaip nurodo pavadinimas, HDL apibūdina aparatūra (angl. *hardware*). Todėl svarbu skaityti ar rašyti kodą atsižvelgiant į aparatūrą ir jos perspektyvas. VHDL leidžia aprašyti bet kokį aparatūriškai realizuojamą algoritmą, todėl VHDL kalba labai panaši į programavimo kalbą. Skirtumas toks, kad VHDL papildomai prie konstantų ir kintamųjų dar turi signalus, su kuriais siejami laiko parametrai ir galima nurodyti operatorių vėlinimus. 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 sintezuojamas. [9] Visuose atvejuose programinė kalba VHDL bus universalesnė.

Ši programinė kalba leidžia aprašyti kaip ir skaitmeninės grandinės darbą, taip ir struktūros tipą. VHDL naudojama daugelyje sistemų, modeliuojant skaitmenines grandines, projektuojant programuojamas logines integralines mikroschemas ir bazinius matricos tipo kristalus.

Pavyzdyje (žr. 5 pav.) pateiktas sumatorius ir jo teisingumo lentelė. *A* ir *b* yra sumatoriaus įvesties bitai, kurie bus sudedami, *cin* yra pernašos įvesties bitas,o išvesties bitai: s – suminis bitas ir *count* yra pernašos išvesties bitas. Kaip galima matyti teisingumo lentelėje, išėjimo bitas *s* turi įgauti vieneto reikšmę, kas kartą, kai įėjimų vienetų suma yra nelyginė, tuo metu *count* turi įgyti vieneto reikšmę, kai du ar daugiau įėjimų įgauna vieneto reikšmę.

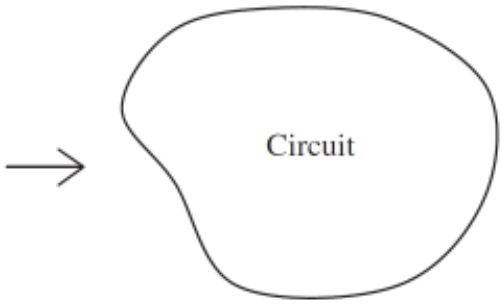


a b cin	s cout
0 0 0	0 0
0 1 0	1 0
1 0 0	1 0
1 1 0	0 1
0 0 1	1 0
0 1 1	0 1
1 0 1	0 1
1 1 1	1 1

5 pav. **Sumatoriaus diagrama ir jo teisingumo lentelė [13]**

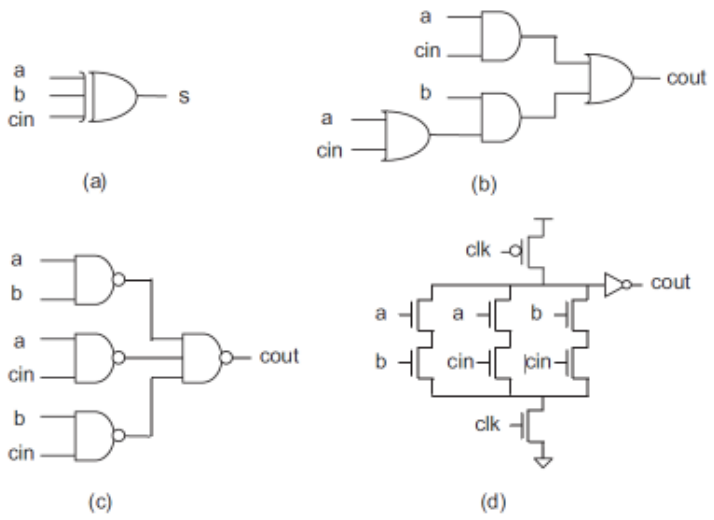
Pateikiamas sumatoriaus VHDL kodas (žr. 6 pav.). Galima matyti, kad kodas prasideda nuo *ENTITY* sakinio. Šis sakinys paskelbia objekto sąsaja su išoriniu pasauliu. Jis nurodo jungties išvadų kiekį, jungties išvadų signalų kryptį, jungties išvadų signalų tipą. *ENTITY* sakinys gali apimti ir daugiau informacijos. Taip pat kode matomas raktažodis *ARCHITECTURE* , tai nurodo, kad šis sakinys aprašys objekto architektūra. Architektūros vardas yra *dataflow,* o objekto kuriam priklauso architektūra vardas yra *full_adder.* Architektūra nusako kaip grandinė turi funkcionuoti. Sekančiose kodo eilutėse aprašoma, kad suminis bitas apskaičiuojamas $s = a \oplus b \oplus cin$, kai *cout* gaunamas skaičiuojant $cout = a.b + a.c + b.cin$.

```
ENTITY full_adder IS
PORT (a, b, cin: IN BIT;
      s, cout: OUT BIT);
END full_adder;
-------------------------------------
ARCHITECTURE dataflow OF full_adder IS
BEGIN
   s <= a XOR b XOR cin;
   cout <= (a AND b) OR (a AND cin) OR
           (b AND cin);
END dataflow;
```



6 pav. **VHDL kodo pavyzdys, sumatorius [13]**

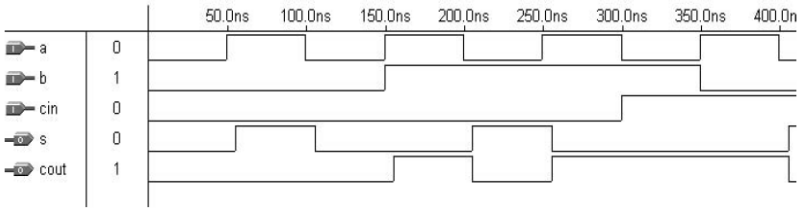
Iš VHDL kodo pateikto paveikslo kairėje (žr. 6 pav.) išvedama grandinė, pateikta paveikslo dešinėje. Yra keli būdai skirtingi realizuoti lygtis aprašytas *ARCHITECTURE* sakinyje, taigi tikroji grandinė priklausys nuo kompiliatoriaus, kuris bus naudojamas ir nuo pasirinktos technologijos. Pateikiami keli grandinių pavyzdžiai, ankščiau pateiktam sumatoriui (žr. 7 pav.). Jeigu projektuotojo tikslas yra programuojamosios logikos prietaisas (PLD arba FPGA), tuomet yra du galimi variantai *cout* išvadui, kurie vaizduojami paveikslo b ir c dalyse (abejuose $cout = a.b + a.c + b.cin$). Kita vertus, jei mūsų tikslas naudoti *ASIC,* tuomet naudojamas *CMOS* realizavimas, tranzistorių lygyje kuris pateikiamas paveikslo d dalyje (kur naudojami *MOS* tranzistoriai ir domino logika). Sintezės įrankis gali būti nustatytas ploto arba greičio optimizavimui, kas akivaizdžiai gali įtakoti galutinę grandinę.



7 pav. **Galimų grandinių pavyzdžiai VHDL kodo pateikto 6 pav. [13]**

Kad ir kokia galutinė grandinė gaunama iš kodo, jos veikimas turi būti patikrintas (testuojamas), kol ji dar yra projektavimo lygyje (po sintezės). Žinoma ji turi būti testuojama fiziniame lygyje, bet tuomet projektavimo keitimai brangiai kainuotų.

Testuojant įrenginį, modeliavimo programoje vaizduojamos banginės diagramos, bus panašios į pateiktas paveiksle (žr. 8 pav.). Paveiksle pateikti VHDL kodo (žr. 5 pav.) modeliavimo rezultatai, kurie apibūdina sumatoriaus (žr. 5 pav.) funkcionalumą. Kaip matoma įėjimo kontaktai (vaizduojami įeinančia rodyklę ir žymimi I simboliu) ir išėjimo kontaktai (vaizduojami įeinančia rodyklę ir žymimi O simboliu) yra tie, kurie išvardinti sumatoriaus VHDL kode *ENTITY* sakinyje. Galima lengvai nustatyti įėjimo signalų reikšmes (šiuo atveju *a, b* ir *cin*) ir modeliavimo programa apskaičiuos ir nubraižys išėjimo signalus (*s* ir *cout*). Iš testo rezultatų, galima matyti, kad išėjimo signalai funkcionuoja taip, kaip tikėtasi.



8 pav. **VHDL kodo (žr. 6 pav.) modeliavimo rezultatai [13]**

Apibendrinant, galima teigti, kad VHDL kalba yra skirta grandinių sintezei ir modeliavimui. Be to, kad VHDL yra pilnai modeliuojama, ne visos konstrukcijos yra sintezuojamos. Pagrindinis motyvuojantis VHDL aspektas yra jos standartiška ir technologiškai nepriklausoma kalba, kuri yra portatyvi ir daugkartinio naudojimo. Taigi VHDL yra priešingybė įprastoms kompiuterinėms programoms, kurios yra nuoseklios, nes šios kalbos komandos yra lygiagrečios. Dėl šios priežasties VHDL yra dažnai nurodoma, kaip kodas, o ne programa. Joje komandos patalpintos sakiniuose *PROCESS, FUNCTION*, arba *PROCEDURE* yra vykdomos nuosekliai.

1.3.2. Verilog

Verilog – aparatūros aprašymo kalba naudojama apibūdinti elektroninę grandinę funkciniu arba struktūrinių aprašymu ir ši kalba yra IEEE standartas. Ši kalba buvo sukurta tuo laiku, kai projektuotojai ieškojo įrankių skirtingo lygio modeliavimo suderinimui. 1980 – aisiais nebuvo įrankių, kurių pagalba galima buvo tai padaryti. Verilog buvo sukurta 1983 Phil Moore, o pirmasis modeliavimas buvo atliktas maždaug po metų. Ji daug pasiskolino iš tuo metu jau egzistuojančių kalbų: lygiagretumo aspektai, sintaksė ir skirtingų lygių derinimas metodai. Oficialiai ši kalba buvo pripažinta 1995 metais, kaip VHDL ja galima aprašyti grandines tiek struktūriniu, tiek elgsenos metodu, naudoti modeliavimui ir sintezei.

Verilog aparatūros aprašymo kalba, skiriasi nuo programų programavimo kalbų, nes ji apima laiko ir signalo priklausomybę. Šioje kalboje yra du priskyrimo operatoriai, bloko priskyrimo (=) ir ne bloko (<=) priskyrimo. Ne bloko priskyrimas projektuotojui leidžia aprašyti būsenos aparatą (angl. *state-machine*) nenaudojant laikinųjų saugojimo kintamųjų (įprastinėse programavimo kalbose, būtina apibrėžti tam tikrus laikino saugojimo vietų operandus, kurie bus vėliau naudojami). Kadangi šios sąvokos yra dalis Verilog kalbos semantikos, projektuotojai gali greitai aprašyti dideles grandines palyginus kompaktiškoje ir glaustoje formoje.

Verilog kūrėjai siekė, kad ši kalba būti panašesnė į C kalbą, kuri yra plačiai naudojama programinės įrangos kūrimo inžinerijoje. Verilog yra viena iš tų kalbų, kurioje: komandų rašymas didžiosiomis ar mažosiomis raidėmis turi reikšmės (angl. *case-sensitive*), turi valdymo srauto raktažodžius (if/else, for, while, case ir pan.), naudojama operacijų tvarka (kartais vadinamas operatorius pirmenybę), naudojama paaiškinti, kurios procedūros turi būti atliekamos pirmiausia atsižvelgiant į matematinė išraišką. Vienas iš sintaksės skirtumų yra kintamųjų deklaracija, Verilog kalboje reikalauja bitų ilgį ar net/reg tipus rašyti [] skliaustuose, o procesinius blokus begin/end {} skliaustuose.

Modulis yra svarbiausia kodo dalis Verilog aparatūros aprašymo kalboje. Tiek elgsenos, tiek struktūrinis aprašymas gali būti pateiktas per modulius. Galima teigti, kad ši kalba sudarytą iš modulių hierarchijos, kurie komunikuoja tarpusavyje, deklaruotų įvesties, išvesties ir dvikrypčiu išvadų pagalba. Viduje modulis gali būti sudarytas iš: kintamųjų deklaracijų (*wire, reg, integer* ir pan.), lygiagrečių ir nuoseklių pareiškimo blokų bei kitų modulių elementų. Nuoseklūs pareiškimai yra rašomi *begin/end* bloke ir vykdomi eilės tvarka. Bet blokai, kurie yra vykdomi lygiagrečiai Verilog atitinka kaip duomenų srauto (angl. *dataflow*) kalba.

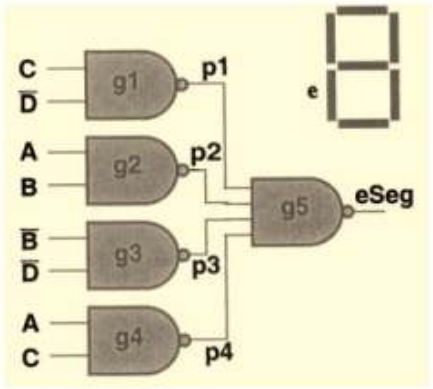
Verilog kalboje signalas gali įgyti keturias reikšmes:

- 0 – loginis nulis arba klaidinga reikšmė;
- 1 – loginis vienetas arba teisinga reikšmė;
- x – nežinoma loginė reikšmė;
- z – pilnutinė varža arba trijų būsenų ventilis.

Ši sistema leidžia abstrakčiai modeliuoti signalus, kai keli šaltiniai sujungti vienu tinklu. Beveik visi Verilog pareiškimai yra sintezuojami, o moduliai atitinkantis sintezuojamą stilių vadinami RTL (angl. *Register Transfer Level*), gali būti fiziškai sintezuojami programinės įrangos. Sintezės programos algoritmiškai transformuoja abstraktų Verilog kodą i tinklo sąrašą (angl. *netlist*), kurį sudaro loginiai elementai (IR, ARBA, NE ir pan.), kurie galimi naudoti FPGA ir VLSI technologijose. Toliau dirbant su *netlist* failu, galima gauti grandinės gamybos planą, pvz *bitstream* failas FPGA programuojamoje matricoje.

Pateikiamas pavyzdys (žr. 9 pav.) loginė schema ir dalis jos kodo. Pavyzdyje matomas septynių segmentų displėjus, vaizduojantis skaičius. Displėjui rodyti skaičius nuo nulio iki devynių ir šešioliktainius skaičius nuo A iki F, reikia į grandinės įėjimo išvadus paduoti 4 bitų ilgio kodą. Verilog kodu pateiktas pavyzdys aprašo E segmento displėjaus atvaizdavimui.

```
1  module binaryToESeg;
2        wire     eSeg, p1, p2, p3, p4;
3        reg      A,B,C,D;
4
5        nand #1
6           g1(p1, C,~D),
7           g2 (p2, A, B),
8           g3 (p3, ~B, ~D),
9           g4 (p4, A, C),
10          g5 (eSeg, p1, p2, p3, p4);
11 endmodule
```



9 pav. **Verilog kalbos pavyzdys 7 segmentų displėjui (E segmentas) [21]**

Iš verilog kodo aprašo galima matyti modulio apibrėžimą, šiuo atveju modulis pavadintas *binaryToESeg*. Kiekvieno modulio apibrėžimui naudojamas raktažodis *module*, po to seka modulio pavadinimas, užbaigiama *endmodule* raktažodžiu. Antroje eilutėje nurodomi laidų (angl. *wire*) pavadinimai, naudojami perduoti logines reikšmes tarp submodulių šiame modulyje.

Trečioje eilutėje paskelbiami saugojimo registrų vardai, kurie saugos reikšmes. Šie registrai yra abstraktūs *flip-flop* elementai. Penktoje eilutėje jos tęsinyje nuo šeštosios iki dešimtosios, aprašomi penki IR-NE ventiliai, kiekvienas iš jų turi vieną laiko vieneto vėlinimą. IR-NE ventiliai - vieni iš išlanksto apibrėžtų loginių elementų, taip pat kaip ir: IR, ARBA, XOR. Šis aprašymas nurodo, kad penki ventiliai pavadinti nuo *g1* iki *g5* egzistuoja grandinėje. Ženklas *#1* nurodo, kad funkcijos bus vykdomos per vieną laiko vienetą. Galiausiai, ženklai skliausteliuose nurodo kaip laidais sujungti registrai ir ventiliai. Pirmasis ženklas skliausteliuose yra ventilio išėjimo, o visi kiti įėjimo išvadai. Ženkliukas „~“ reiškia neigimą. Verilog kodo schematinis vaizdas yra pateiktas, dešinėje paveikslo pusėje (žr. 10 pav.), kad parodyti kodo ir loginės diagramos ekvivalentiškumą.

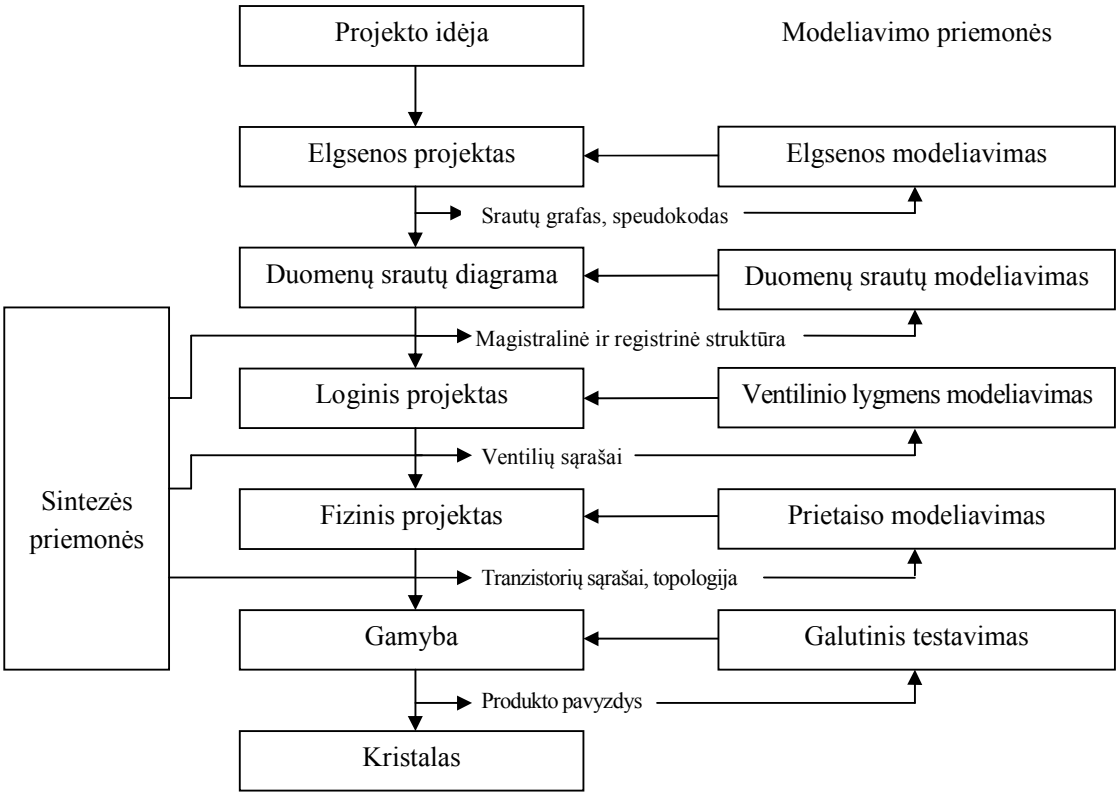
1.4. Skaitmeninių sistemų projektavimas

Projektavimo procesas prasideda nuo idėjos, ir visi kiti darbai išplėtojami iš jos (žr. 10 pav.). Sekantis etapas tai elgsenos aprašymas, kurio metu projektuotojas sudaro diagramas, duomenų srautų grafus ar speudokodus. Šis aprašymo lygmuo abstrakčiausias, aprašomos projekto funkcijos, tačiau nenurodoma kaip jos turėtų būti įgyvendintos. Šio lygmens aprašas gali būti naudojamas kaip projekto dokumentacija ir suprantamas ne tik specialistui, bet ir paprastam vartotojui. Toliau ne ką svarbesnis etapas sudaryti duomenų srautus, kurio metu projektuotojas apibrėžia registrinius ir loginius vienetus, reikalingus sistemos įgyvendinimui. Šios komponentės gali būti sujungtos tarpusavyje, vartojant dvikryptes ir vienkryptes magistrales. Sudaroma testų procedūra, siekiant patikrinti duomenų judėjimą per magistrales tarp registrų ir loginių vienetų. Šiame lygmenyje projektuotojas gali pateikti nurodymus sintezės programai, kad būtų sintezuojamas žemesnio lygmens aprašymas. Loginis projektas - sekantis projektavimo proceso etapas, kurio metu realizuojami registrai, loginiai vienetai, magistralės, parenkami trigeriai bei ventiliai ir tokio lygmens aprašymas dažnai vadinamas projekto struktūriniu aprašymu. Po šio skaitmeninės sistemos projektavimo proceso etapo – nesudėtinga sintezuoti aparatūrą. Sekantis etapas transformuoja prieš tai buvusio etapo ventilių sąrašą tranzistorių sąrašą arba topologiją, procedūros metu trigeriai ir ventiliai pakeičiami jų tranzistoriniais ekvivalentais arba bibliotekos ląstelėmis. Paskutinis projektavimo etapas yra gamyba, kurios metu naudojamas tranzistorių sąrašas arba topologijos specifikaciją, kad pagaminti kristalą.

Beveik visi šie etapai šiomis dienomis gali būti atlikti kompiuterinėmis priemonėmis, taip labai palengvindamos projektuotojui darbą. Dauguma priemonių gali aprašyti projektą, generuoti aparatūrą, generuoti testines sekas, atlikti modeliavimą ir sintezę.

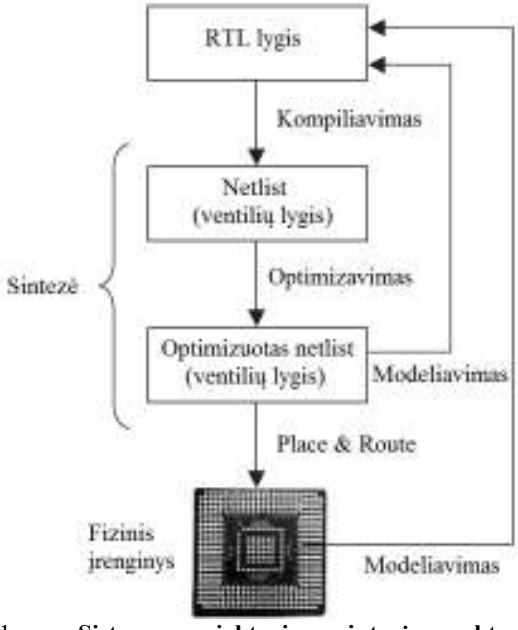
Pateiktoje sistemos projektavimo procesų sekoje galima matyti, kad modeliavimo programa gali būti taikoma kiekviename projektavimo etape. Be schemos aprašymo programai dar reikia modeliavimo duomenų ir testinių rinkinių. Taikydama testinius rinkinius programa priskiria reikšmes įėjimams ir sumodeliuoja schemos reakcijas. Visus rezultatus projektuotojas gali stebėti pateiktus lentelėmis arba laikinėmis diagramomis. Aukštesniuose projektavimo proceso etapuose modeliavimas pateikia informaciją apie projektuojamos schemos funkcionalumą.

Dar viena projektavimo priemonė, automatiškai transformuojanti projekto aprašymą iš vienos formos į kitą, vadinama sinteze. Sintezės priemonės padeda projektuotojui keliuose projekto etapuose.



10 pav. **Skaitmeninės sistemos projektavimo procesas įtraukiant modeliavimą ir sintezę.[9]**

Kaip jau minėta ankščiau, vienas svarbiausių aparatūros aprašymo kalbų privalumas – galimybė sintezuoti grandinę arba sistemą į programuojamą įrenginį (PLD arba FPGA). Norint atlikti sintezę, reikia sekti etapų, kurie pavaizduoti 11 paveiksle. Projektas pradedamas rašyti aparatūros aprašymo kalbos kodu, ir išsaugotu faile su atitinkama galūne (.vhd, .v, .h, .cc) priklausomai nuo pasirinktos kalbos.



11 pav. **Sistemos projektavimas sintezės aspektu [13]**

Pirmas sintezės žingsnis yra kompiliavimas. Jis naudojamas konvertuoti aukšto lygio kalbą, kuri aprašo RTL (angl. *Register Transfer Level*) į tinklo sąrašą (angl. *netlist*) ventilių lygyje. Kitas žingsnis yra optimizavimas, kuris yra vykdomas ventilių lygyje ir optimizuojamas greitis bei plotas. Šiame etape jau galimas modeliavimas. Pabaigoje *plane & route* programa negeruoja fizinę schemą (angl. *layout*) PLD arba FPGA sistemai.

1.5. Projektavimo erdvės apžvalga

Dažniausiai projektavimui yra naudojami mikrovaldikliai bei programuojamos loginės matricos. Mikrovaldikliai nuo programuojamų loginių matricų skiriasi tuo, kad juose iš karto mikrograndyno gamybos metu suformuojami loginiai elementai, kurių paskirtis bei atliekamos loginės operacijos niekada nesikeičia. Šiuolaikiniai mikrovaldikliai yra kur kas tobulesni, lyginant gamintais prieš dešimtmetį, bet jų veikimas, iš principo, paremtas tais pačiais metodais, sukurtais prieš keliasdešimtį metų. Esmė yra ta, kad mikrovaldikliams rašomos programos – komandų seka, kurios tiksliai nurodo ką procesorius turi daryti, todėl procesorius visą laiką siunčiasi komandas iš operatyvinės atminties. Iš dalies tai skaitmeninių procesorių privalumas, kadangi galima realizuoti bet kokio sudėtingumo logines operacijas ar kitokius veiksmus, tai lėmė jų paplitimą ir taikymą praktikoje. Sprendžiant sudėtingus (specializuotus) uždavinius, mikrovaldiklių efektyvumas yra nepakankamas. Tam geriau tinka DSP (angl. – *Digital Signal Processor*), bet visiems gyvenimo atvejams per brangu gaminti specializuotus lustus. Dėlto buvo sukurtos programuojamos loginės matricos[8].

1.5.1. Mikrovaldikliai

Mikrovaldiklis yra mikroschema skirta valdyti elektroninius įrenginius, kuri sudaryta iš procesoriaus branduolio, atminties ir programuojamų įėjimo ir įėjimo išvadų. Programos atmintis tai pat dažnai įtraukta į lustą, priklausomai nuo atminties kiekio. Mikrovaldikliai suprojektuoti įterptinėms sistemoms, skirtingai nei mikroprocesoriai, kurie naudojami personaliniuose kompiuteriuose.

Mikrovaldikliai naudojami įrenginiuose turinčiuose automatinį valdymą, pavyzdžiui automobilio variklio kontrolės sistema, implantuojamuose medicinos įrenginiuose, valdymo pultuose, buitiniuose prietaisuose, elektriniuose įrankiuose, žaisluose ir t.t. Naujos technologijos leidžia projektuoti vis mažesnes ir pigesnes sistemas. Todėl sistemos kurios, naudoją atskirą mikroprocesorių, atmintį, įėjimo ir išėjimo įrenginius vis rečiau naudojamos, nes mikrovaldikliai atlieka tas pačias funkcijas už mažesnę kainą ir sutaupo schemose vietos. Nauji mikrovaldikliai palaiko mišrius sinaglus (skaitmeninius ir analoginius), tai pravartu sistemose kur naudojami analoginės elektrinės sistemos. Taigi paprastai mikrovaldikliai naudojami ten, kur stengiamasi sumažinti lustų skaičių, bei sąnaudas, bet reikalingas lankstumas.

Mikrovaldiklių architektūra

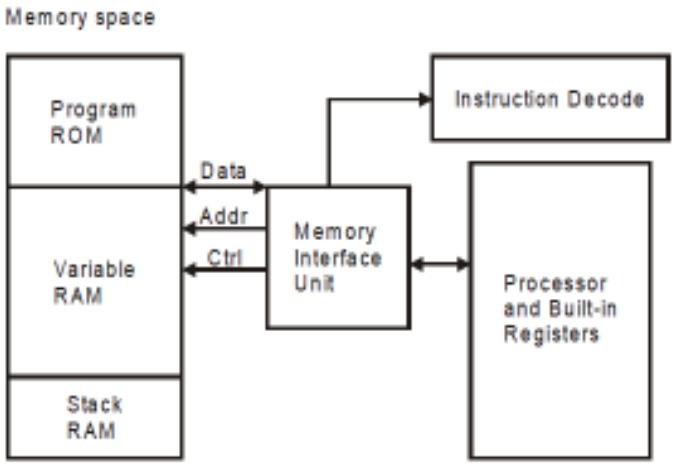
Mikrovaldiklių rinka labai plati, todėl nesunku surasti projekto reikalavimus geriausiai atitinkantį produktą. Mikrovaldikliai pagal architektūrą skiriami į dvi grupes:

- Von-Neuman (arba Princeton)
- Harvard

Šios dvi architektūros skiriasi tuo, kad jose programos kodas ir duomenys skirtingai saugomi, bei pasiekiami. Harvard ir Von-Neuman. Harvard būdingos dvi skirtingos magistralės, duomenų ir programos, tuo tarpu Von-Neuman architektūroje ji yra bendra. Dėl šios priežasties dažnai Harvard architektūra yra pranašesnė greitaveikos atžvilgiu.

Von-Neuman architektūra

Mikrovaldikliuose, kuriuose naudojama Von-Neuman architektūra (žr. 12 pav.), turinti vieną duomenų magistralę instrukcijoms ir duomenims. Programos instrukcijos ir duomenys saugojami bendroje pagrindinėje atmintyje. Kai tokios architektūros valdiklis siunčia adresą atminties blokui, pirmiausia nuskaitomi nudarymai ir tik tada nuskaitomi duomenys patvirtinantys instrukciją. Šie du atskiri veiksmai lėtina mikrovaldiklio veikimą, bet tai nėra esminė priežastis, kuri lemią valdiklio pasirinkimą.



12 pav. **1 pav. Von-Neuman architektūros blokinė diagrama [4]**

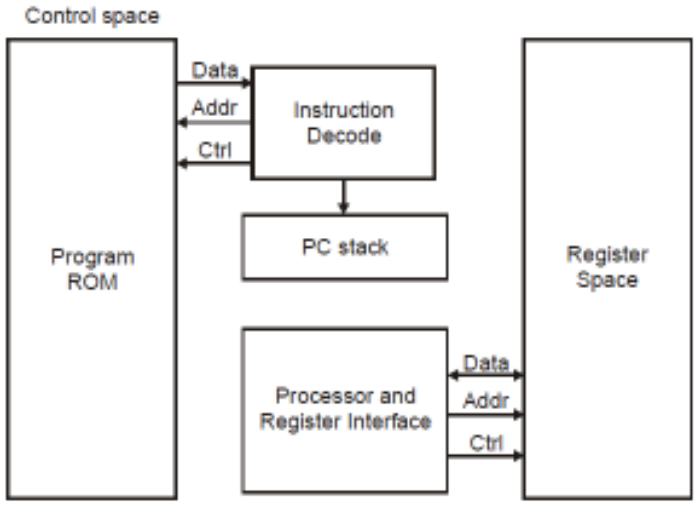
Von-Neuman architektūros pagrindinis privalumas, kad ji supaprastina mikrovaldiklio veikimą, nes naudojama yra tik viena atmintis. Duomenų ir programos instrukcijų saugojimui mikrovaldikliuose naudojama adresuojamoji atmintis – RAM (angl. *Random Access Memory*).

Kaip pavyzdys: Motorolos kompanijoje apgamintas 68HC11 mikrovaldiklis su Von-Neuman architektūra. Norint nuskaityti baitą iš atminties ir išsaugoti jį kaupiklyje reikia įvykdyti šiuos ciklus:

- Ciklas 1: Instrukcijos nuskaitymas
- Ciklas 2: Nuskaitomi duomenys iš RAM ir patalpinami kaupiklyje

Harvard architektūra

Mikrovaldikliuose, kuriuose naudojama Harvard architektūra (žr. 13 pav.), turinti atskiras duomenų ir instrukcijų magistrales. Tai leidžia atlikti lygiagrečius veiksmus. Tuo metu kai instrukcijos yra nuskaitomos, duomenų magistralėje jau gaunami duomenys į siunčiamas instrukcijas. Kai viena instrukcija yra baigta, kita jau paruošta siuntimui. Toks duomenų dalijimasis teoriškai leidžia žymiai didesni perdavimo greitį, nei naudojant Von-Neuman architektūrą.



13 pav. **Harvard architektūros blokinė diagrama [4]**

Mikrovaldikliuse su Harvard architektūra įvykdyti instrukcijai reikalinga mažiau ciklų nei Von-Neuman architektūroje. Pavyzdžiui Intel komanijos MCS-51 šeimos, PIC ar AVR mikrovaldikliai naudoja Harvard arba šiek tiek modifikuota architektūra.

Kaip ir prieš tai aptartoje architektūroje, taip ir šioje pateikiamas pavyzdys baito nuskaitymui:

- Ciklas 1: Užbaigta ankstesnė instrukcija
  Nuskaitoma instrukcija „Perkelti duomenis į kaupiklį“
- Ciklas 2: Įvykdoma instrukcija „Perkelti duomenis į kaupiklį“
  Nuskaitoma sekanti instrukcija

CISC architektūros mirkovaldikliai

Dauguma šiuolaikinių mikrovaldiklių yra sukurti šios architektūros pagrindu. Jei mikrovaldiklis turi instrukcijų rinkinį, kuris gali palaikyti keletą adresavimo būdų, skirtų aritmetinėms ir loginėms instrukcijoms, duomenų perdavimo ir atminties pasiekimo instrukcijoms, tuomet mikrovaldiklio architektūra gali būti apibrėžta kaip CISC,

Tipinis CISC mikrovaldiklis turi daugiau nei 80 skirtingų instrukcijų, kai kurios iš jų labai galingos, kai tuo tarpu kitos specializuotos specifinei užduočiai.

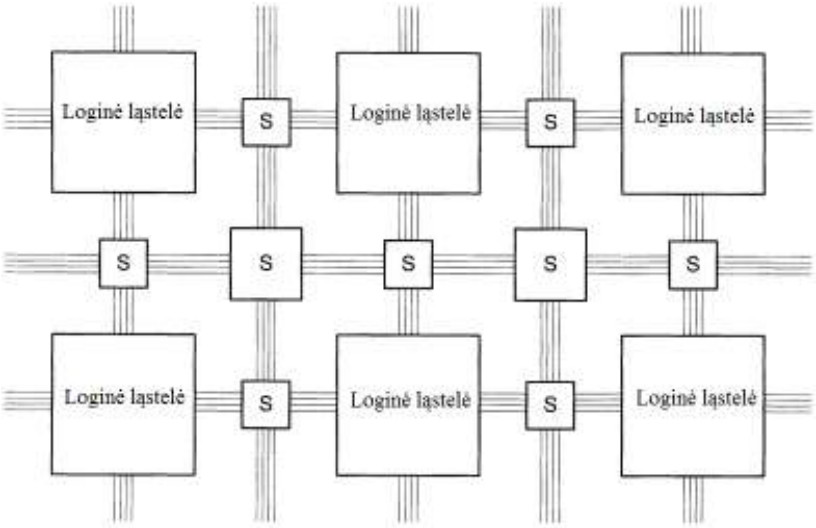
CISC architektūros privalumas - dauguma instrukcijų yra plataus profilio, tai leidžia programuotojui naudoti vieną vietoje kelių instrukcijų. Kitais žodžiais kiekviena instrukcija gali įvykdyti keletą žemo lygio operacijų, tokių kaip užkrovimas iš atminties, aritmetinė operacija ir įrašymas į atmintį, viską atliekant viena instrukcija.

RISC architektūra

RISC – tai mikrovaldiklių instrukcijų rinkinio architektūra, kuri sudaryta iš elementarių instrukcijų, kurios atlieka po viena operaciją. Šios architektūros atsiradimą įtakojo supratimas, kad daugelis ypatybių, kurios buvo įtraukiamos į tradicinius valdiklių projektus siekiant lengvesnio programavimo, buvo pradėtos ignoruoti. RISC architektūrą palyginus su CISC - procesoriaus komandų daug mažiau, jos paprastesnės, dėl to jas įvykdyti užima mažiau centrinio procesoriaus ciklų. Kadangi nereikia kurti sudėtingo komandų rinkinio, galima tiksliai prognozuoti komandų vykdymo laiką (procesoriaus ciklais), inžinieriams lieka daugiau galimybių optimizuoti procesorius, ypač – konvejerinį komandų vykdymą. RISC komandos paprastai būna vieno ilgio, kiekviena komanda turi tik vieną arba du operandus. RISC procesoriuose naudojama daug bendrosios paskirties registrų, sudarančių registrų langus. Adresavimo galimybės tokiuose procesoriuose paprastai būna smarkiai apribojamos: esant reikalui, tam tikras atminties puslapis perkeliamas į registrų langą arba atvirkščiai, tačiau komandų, tiesiogiai dirbančių su atmintyje esančiais duomenimis, nėra.

1.5.2. FPGA programuojamos matricos

FPGA programuojamoji matrica yra loginis įrenginys, turintis dvimačio masyvo tipo loginių ląstelių ir programuojamus jungiklius (paveiksle pažymėti S). FPGA konceptuali struktūra pavaizduota 14 paveiksle. Loginės ląstelės gali būti konfigūruojamos (programuojamas) siekiant atlikti norimą funkciją, o programuojami jungikliai pritaikomi teikti sujungimus tarp loginių ląstelių. Projektavimas vykdomas nurodant funkcijas kiekvienai loginei ląstelei ir pasirinktinai nustatant programuojamų jungiklių sujungimus. Kai projektavimas ir sintezė atlikti, paprastu duomenų kabeliu galima nusiųsti norimą loginę ląstelę į FPGA lustą ir gauti išskirtinę grandinę. Taigi šie veiksmai gali būti atliekami, nebe gamykloje ar laboratorijoje,o net ir namuose naudojant programuojamąją matricą[15].



14 pav. **Konceptuali FPGA lusto struktūra [15]**

Rinkoje pasirodžius FPGA (Field Programmable Gate Array), atsirado galimybė projektuoti mažesnių matmenų spausdintines plokštes. Vis didėjanti programuojamojo lauko apimtis, universalumas ir sparčiai mažėjanti kaina didina šio tipo schemų populiarumą ir paplitimą. Anksčiau norimai loginei funkcijai atlikti reikėdavo keleto ar keliolikos lustų, o dabar naudojamas vienas FPGA lustas. Programuojamosios logikos lustas gali būti pasirinktas priklausomai nuo reikiamo jo dydžio: loginių elementų ir trigerių skaičiaus; reikiamos maitinimo ir signalų įtampos (1,5; 3,5; 5V).

Šio tipo lustų naudojimo teigiamybės:

1. Sutaupoma nemažai spausdintinės plokštės ploto, nes sudėtingą skaitmeninę projekto dalį galima gauti naudojant vieną lustą. Taip suprojektuojama kompaktiška spausdintinė plokštė su viena logine mikroschema;

2. Galima sumažinti projektavimo trukmę, kadangi spausdintinė plokštė gali būti projektuojama ir gaminama lygiagrečiai su skaitmeninės dalies projektavimu;

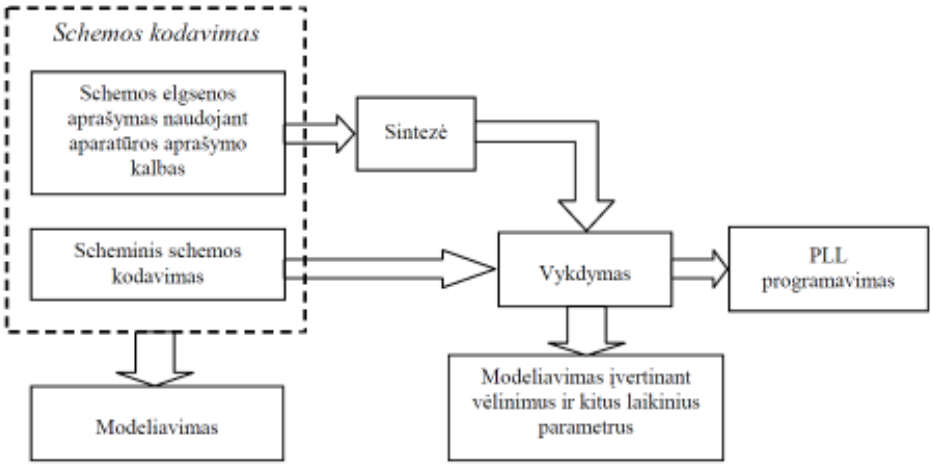
3. Pagamintos plokštės atliekamą funkciją galima lengvai pakeisti perprogramavus FPGA atminties lustą, tam nereikia jos iš naujo projektuoti ir gaminti. Padarius klaidą, užtenka perprogramuoti lustą. Kiekviename projektavimo etape galima keisti schemos funkciją;

4. Galima padidinti schemos veikimo taktinį dažnį ir apdorojamų duomenų srautą, tai pat paspartinti ir supaprastinti skaitmeninio įtaiso projektavimą panaudojant jau sukurtus tipinių funkcinių mazgų bibliotekos elementus;

5. Projektuotojo kūrinys yra saugus, niekas negali jo nukopijuoti.

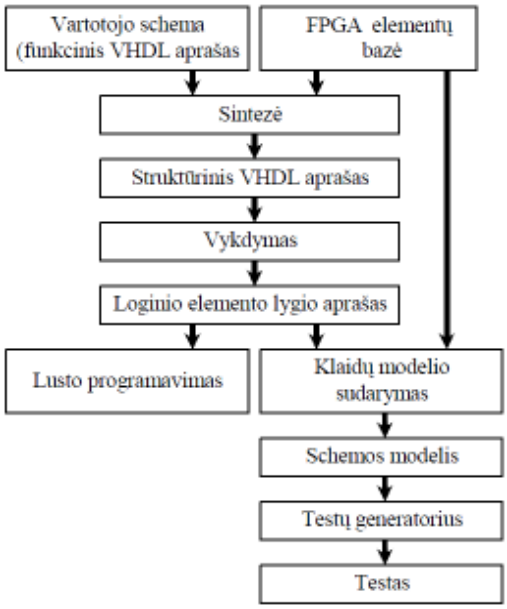
Šiuolaikinių elektroninių schemų projektavimas jau nėra susijęs su konkrečia elementų baze. Programinės priemonės tarsi suvienodina įvairias programuojamųjų struktūrų architektūras. Tik paskutiniame projektavimo etape pasirenkamas konkretus lustas ir jo elementų bazė. Programuojamosios logikos matricų gamintojų tiekiamos programinės įrangos pakanka

visai schemos projektavimo eigai, pradedant schemos kodavimu ar elgsenos aprašymu ir baigiant modeliavimu, mikroschemos programavimu ir verifikavimu.



15 pav. **Projektavimo procesas naudojant programuojamosios logikos matricas [1]**

Jei schema yra didesnė arba žinoma tik jos elgsena, t.y. žinomi tik įėjimo signalai ir schemos reakcija į juos, geriau naudoti aparatūros aprašymo kalbas: VHDL arba Verilog. Toks loginės schemos aprašas sintezuojamas pasirinktoje FPGA lustų šeimų elementų bazėje. Taip gaunamas schemos aprašas, kurį galima naudoti programuojant PLL. Projektavimo, panaudojant PLL, procesas pavaizduotas 15 pav.



16 pav. **Testų generavimo proceso eiga [1]**

Po pagaminimo programuojamasis lustas turi būti visiškai patikrintas nepriklausomai nuo to, kokia konfigūracija vėliau bus naudojama. Siekiant paspartinti patikrą, testinių sekų generatoriai turi sugeneruoti kuo mažiau ir kuo trumpesnes testines sekas, patikrinančias visą programuojamąjį lustą (žr. 16 pav.). Vartotojo testinės procedūros tikslas visiškai kitoks: šiuo

atveju nereikia patikrinti viso programuojamojo lusto, užtenka patikrinti tik tą dalį, kurią naudoja konkreti konfigūracija. Todėl šis uždavinys supanašėja į tradicinį ASIC schemų testų generavimą.

Pagrindiniai FPGA privalumai – greitis ir lankstumas, todėl teisingai panaudojus šias savybes galima pasiekti gerų rezultatų. Yra trys pagrindiniai faktoriai susiję su greičiu: pralaidumas, laukimas (angl. *latency*), ir laikas. Atsižvelgiant į duomenų apdorojimą FPGA luste, pralaidumas reiškia duomenų kiekį, kurie būtų apdorojami per takto ciklą (dažnas matavimo vienetas – bitai per sekundę). Laukimas reiškia laiko trukmę tarp duomenų įvesties ir apdorojamų išvesties duomenų (matavimo vienetas laikas ir takto ciklas). Laikas nurodo vėlinimų logiką tarp nuosekliųjų elementų. Kai sakoma projektas „neatitinka laiko“, turima galvoje, kad kritinis vėlavimo kelias, kuris yra didžiausias vėlinimas tarp *flip-flop* elementų (kuriuos sudaro kombinatoriniai vėlinimai, *clk-to-out* vėlinimai, maršruto vėlinimai, laiko nustatymai ir t.t.) yra didesnis nei taktiko laikrodžio periodas (angl. *clock period*). Standartiniai laiko matavimo vienetai yra taktinio laikrodžio periodas ir dažnis.

2. FAILINIŲ SISTEMŲ REALIZACIJŲ SKIRTŲ MIKROVALDIKLIAMS TYRIMAS

Failinių sistemų realizacijų skirtų mikrovaldikliams tyrimui buvo pasirinktos 5 realizacijos. Visos jos laisvai prieinamos internetinėje erdvėje, dažniausiai atviro kodo ir be veiksmus ribojančių licenzijų. Tyrimo metu siekiama išnagrinėti pagrindines realizacijų funkcijas, jų pritaikymą, specifines savybes bei svarbiausią tokio tipo sistemų privalumą – duomenų perdavimo greitį. Kiekvienai realizacijai kuriama lentelė, kurioje pateikiamas failinės sistemos realizacijos pavadinimas, autorius, testavimui naudojamos elektroninės duomenų saugojimo laikmenos tipas, naudojamas mikrovaldiklis, mikrovaldiklio veikimo taktinis dažnis, naudojama failinė sistema bei pasiekti duomenų rašymo/skaitymo greičiai su skirtingų dydžių failais.

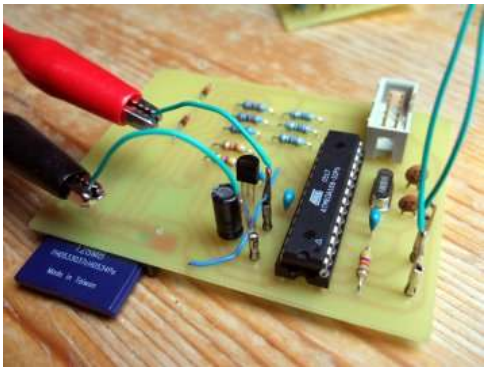
Pirmoji tiriama Roland Riegel failnės sistemos realizacija ir pateikiami jos duomenys.

4 lentelė. Roland Riegel realizacijos pagrindiniai duomenys

Pavadinimas	MMC/SD/SDHC card library			
Kūrėjas	Roland Riegel			
Realizacijos prieiga per internetą	http://www.roland-riegel.de/sd-reader/			
Testavimui naudojama laikmena	128 MB SD kortelė		4 GB SDHC kortelė	
Realizacijoje naudojamas mikrovaldiklis	ATmega128		ATmega128	
Realizacijos veikimui naudojamas dažnis (MHz)	14.745		14.745	
Realizacijoje galimos naudoti FAT failinės sistemos	FAT16		FAT32	
Pasiektas maksimalus duomenų perdavimo greitis (Įrašymas)	Failo dydis (B)	Perdavimo greitis(kB/s)	Failo dydis (B)	Perdavimo greitis(kB/s)
	2048	29,3	2048	73,1
	1024	29,3	1024	73,1
	512	29,3	512	73,1
	64	25,8	64	51,2
	8	23,0	8	28,1
Pasiektas maksimalus duomenų perdavimo greitis (Skaitymas)	Failo dydis (B)	Perdavimo greitis(kB/s)	Failo dydis (B)	Perdavimo greitis(kB/s)
	2048	204,8	2048	240,9
	1024	204,8	1024	240,9
	512	204,8	512	227,6
	64	178,1	64	157,5
	8	102,4	8	43,6

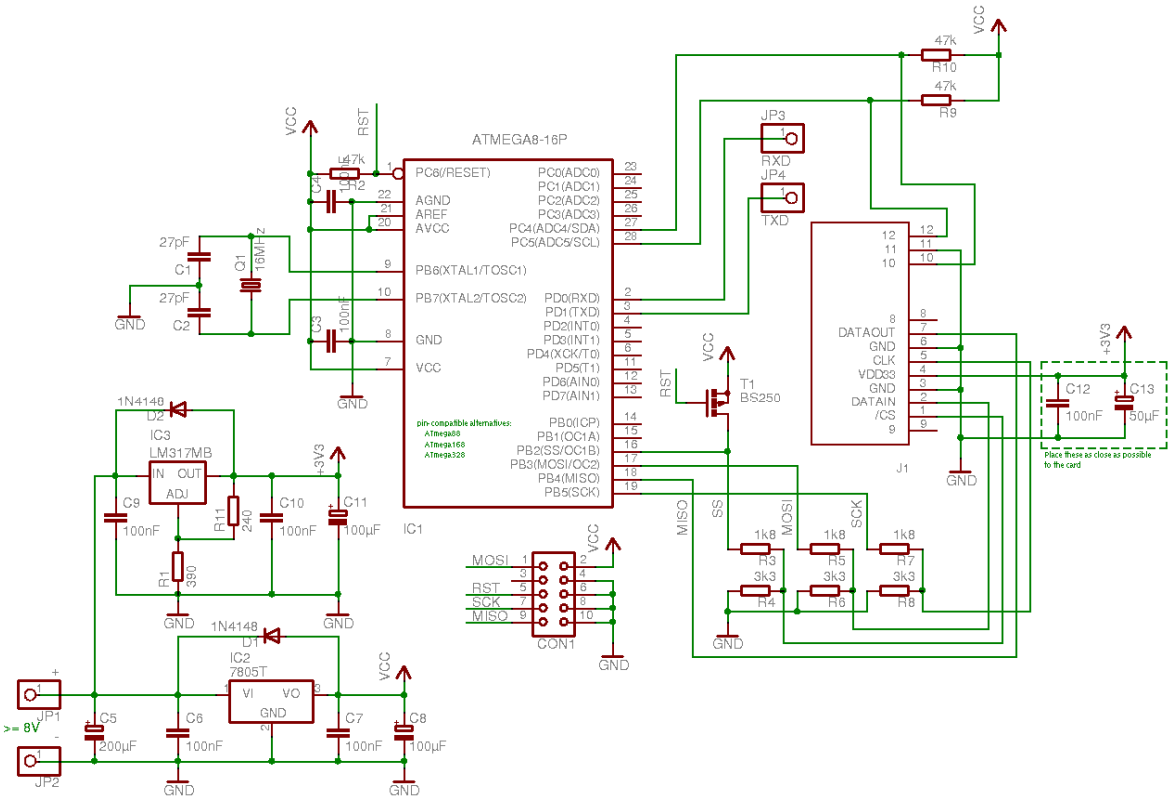
Iš 4 lentelės matoma kad sistema buvo testuojama naudojant ATmega128 mikrovaldiklį, kurio taktinis dažnis 14,745. 128 MB SD ir 4 GB SDHC kortelės testuojant buvo suformatuotos FAT16 ir FAT32 sistemomis. Iš gautų duomenų perdavimo rezultatų matyti, kad greitis palaipsniui kylą didinant failą, tačiau naudojant 512 baitų ir didesnius failus, perdavimo greitis išlieka toks pat.

Kūrėjas savo projekte naudoja skirtingus AVR šeimos mikrovaldiklius (ATmega8, ATmega168, ATmega328) priklausomai nuo atminties kiekio valdikliuose (žr. 17 pav.).



17 pav. **Roland Riegel sukurta realizacija skirta mikrovaldikliui [16]**

Roland Riegel sukurta failinių sistemų realizacija skirta Atemel AVR šeimos mikrovaldikliams. Realizacijos schemoje (žr. 18 pav.) galima matyti naudojamus komponentus.



18 pav. **Roland Riegel sukurtos realizacijos skirtos mikrovaldikliui schema [16]**

Sukurta realizacija parašyta C kalba, projektuotojo žodžiais nėra pati mažiausia ir greičiausia, bet gana lanksti. Naudojama komandinė sąsaja naudojant *rs232 UART 9600 bound rate*. Yra sukurtos komandos, panašias į naudojamas *UNIX* terminale, kuriomis vartotojas valdo mikrovaldiklį (komandos: *cat, cd, disk, init, rite ir pan.*) Realizacijos kodas užima 10794 baitus, taip pat paskaičiuota kad naudojantis realizacija su FAT16 failinė sistema programa užima 728 baitus atminties. Duomenys perduodami naudojant SPI sąsają.

Projektuotojas viešai leidžia naudotis savo kūriniu, beje yra galimybė pritaikyti jį kitos šeimos mikrovaldikliams redaguojant *makefile* failą. Svetainė yra reguliariai atnaujinama, taisomos realizacijos klaidos atsižvelgiant į naudotojų pastabas.

Antra tiriama FatFs failinės sistemos realizacija ir pateikiami jos duomenys.

5 lentelė. FatFs failinės sistemos realizacijos pagrindiniai duomenys

Pavadinimas	Fatfs generic fat file system module			
Kūrėjas	Nežinomas inžinierius iš Japonijos			
Realizacijos prieiga per internetą	http://elm-chan.org/fsw/ff/00index_e.html			
Testavimui naudojama laikmena	512 MB Panasonic SD kortelė		40 GB Toshiba HDD	
Realizacijoje naudojamas mikrovaldiklis	ATmega64		ATmega64	
Realizacijos veikimui naudojamas dažnis (MHz)	9,2		9,2	
Realizacijoje naudojama FAT failinė sistema	FAT32		FAT32	
Pasiektas maksimalus duomenų perdavimo greitis (Įrašymas)	Failo dydis (B)	Perdavimo greitis(kB/s)	Failo dydis (B)	Perdavimo greitis(kB/s)
	2048	27	2048	994
	100	25	100	349
	1	13	1	21
Pasiektas maksimalus duomenų perdavimo greitis (Skaitymas)	Failo dydis (B)	Perdavimo greitis(kB/s)	Failo dydis (B)	Perdavimo greitis(kB/s)
	2048	236	2048	1547
	100	207	100	553
	1	20	1	21

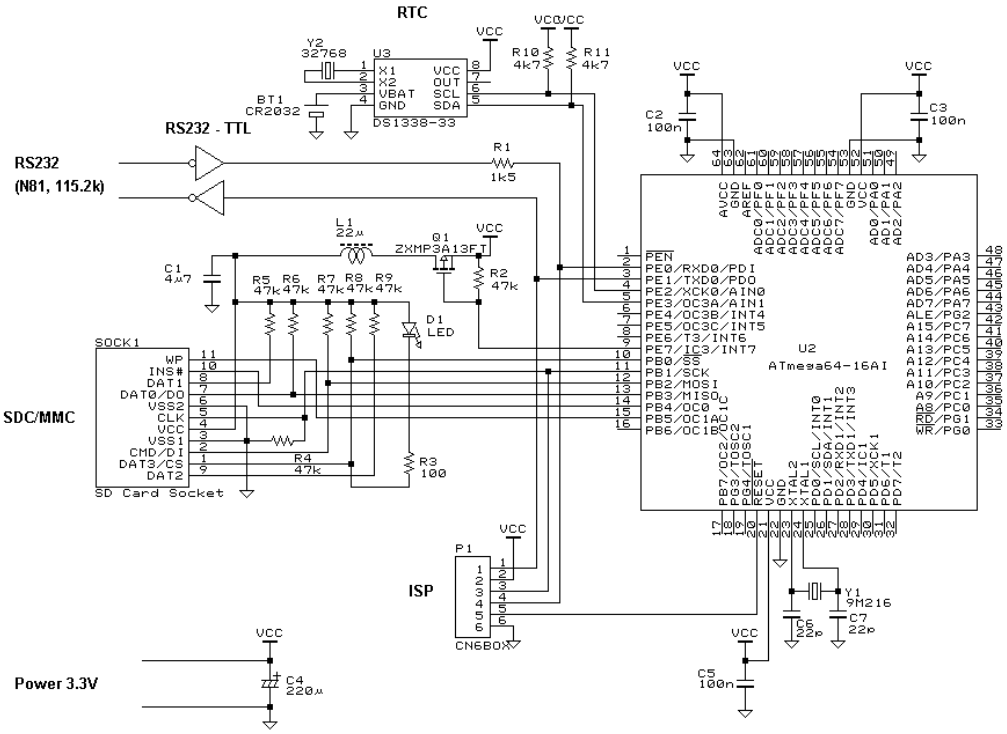
Iš pateiktų duomenų 5 lentelėje galima matyti, kad realizacijos testavime buvo naudoti trejų dydžių failai duomenų perdavimo greičio matavimui FAT32 failinėje sistemoje. Naudojant 1 baitą siuntimui, beveik visomis sąlygomis gautas apytiksliai 20 kB/s greitis, greitis nėra mažas, tiesiog esant tokiam mažam failo dydžiui, greičiui nėra galimybės pakilti per trumpą laiko tarpą. Naudojant SD kortelę duomenų perdavimo greitis gana panašus į Roland Riegel realizacijos (žr. 4 lentelė). Tačiau antra testavimo dalis atlikti naudojant standųjį diską, todėl greitis žymiai didesnis nei SD kortelėje (944 kB/s įrašant 2 MB failą ir 1547 kB/s skaitant 2 MB failą).

FatFs – failinių sistemų realizacija, skirta FAT12,FAT16,FAT32 sistemoms. Kūrėjas prie mikrovaldiklio valdomos sistemos sugebėjo prijungti standųjį diską ir SD korteles bei pasiekti didelį greitį. Tinklapyje pateikiama informacija, kad realizacija parašyta C programavimo kalba ir gali būti naudojama be didelių pakeitimų su įvairiais mikrovaldikliais (AVR, 8051, PIC, ARM, Z80, 68k ir pan.). Duomenys šioje sistemoje perduodami naudojant SD magistralės sąsają.

Šios sistemos privalumai:

- Platformos nepriklausomumas
- FAT12,FAT16,FAT32 palaikymas
- Mažas programos kodas
- Sistemos lankstumas
    - Skirtingų laikmenų palaikymas (MMC,SD, HHD)
    - Ilgų failų pavadinimų palaikymas
    - RTOS (angl. *Real-time operating system*) palaikymas
    - Skirtingų sektorių dydžių palaikymas

Realizacijoje sukurtas komandų sąrašas, kuriomis valdoma failinė sistema (*f_open, f_close, f_mkdir* ir pan.). Sistema naudojant AVR mikrovaldiklį užima 12646 baitų vietos.



19 pav. **FatFs failinės sistemos realizacijos schema ATmega64 mikrovaldikliui [6]**

19 paveiksle pateikiama FatFs failinės sistemos realizacijos ATmega64 mikrovaldikliui schema, bei naudojami komponentai.

Apibendrinant – tai nemokama failinės sistemos realizacija pritaikyta mikrovaldikliui, kuria leidžiama naudotis švietimo, ir mokslinių tyrimų tikslams. Projektas saugomasGNU GPL licenzijos, todėl galima laisvai keisti programos kodą ir pritaikyti pagal poreikius. Tinklapis reguliariai atnaujinamas, kuriamos naujos sistemos versijos atsižvelgiant i vartotojų pastabas.

Trečia tiriama SDFileSystem failinės sistemos realizacija ir pateikiami jos duomenys.

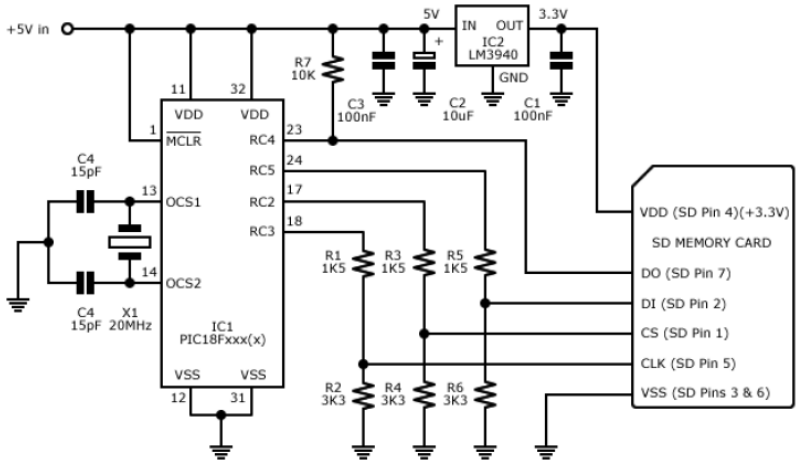
6 lentelė. SDFileSystem library failinės sistemos realizacijos pagrindiniai duomenys

Pavadinimas	Sdfilesystem library			
Kūrėjas	Swordfish komanda			
Realizacijos prieiga per internetą	http://www.sfcompiler.co.uk/wiki/pmwiki.php?n=SwordfishUser.SDFileSystemVersion			
Testavimui naudojama laikmena	IT Works SD 64 MB		IT Works SD 64 MB	
Realizacijoje naudojamas mikrovaldiklis	PIC18F		PIC18F	
Realizacijos veikimui naudojamas dažnis (MHz)	10		40	
Realizacijoje naudojama FAT failinė sistema	FAT16		FAT16	
Perdavimo magistralė	**SPI realizuota programiškai**			
Pasiektas maksimalus duomenų perdavimo greitis (Įrašymas)	Failo dydis (B)	Perdavimo greitis(kB/s)	Failo dydis (B)	Perdavimo greitis(kB/s)
	1024	7,95	1024	31,17
Pasiektas maksimalus duomenų perdavimo greitis (Skaitymas)	Failo dydis (B)	Perdavimo greitis(kB/s)	Failo dydis (B)	Perdavimo greitis(kB/s)
	1024	11,10	1024	43,08
Perdavimo magistralė	**SPI realizuota aparatūriškai**			
Pasiektas maksimalus duomenų perdavimo greitis (Įrašymas)	Failo dydis (B)	Perdavimo greitis(kB/s)	Failo dydis (B)	Perdavimo greitis(kB/s)
	1024	16,82	1024	64,48
Pasiektas maksimalus duomenų perdavimo greitis (Skaitymas)	Failo dydis (B)	Perdavimo greitis(kB/s)	Failo dydis (B)	Perdavimo greitis(kB/s)
	1024	23	1024	86,49

Iš 6 lentelėje pateiktų duomenų matoma kad failinės sistemos realizacija pritaikyta PIC18f mikrovaldikliui. Testavimui pasirinkta viena 64 MB SD kortelė suformatuota naudojant FAT16 failinę sistemą. Ši realizacija buvo testuojama naudojant 10 MHz ir 40MHz mikrovaldiklio dažnį bei aparatūrinę ir programinę SPI sąsaja, taigi duomenų perdavimo greičiai atitinkamai skiriasi. Naudojant programiškai realizuotą SPI rašymo greitis: 7,95 ir 31,17 kB/s, o rašymo greitis 11,10 ir 43,08 kB/s priklausomai nuo naudojamo dažnio. Naudojant aparatūriškai realizuotą SPI skaitymo greitis: 23 ir 86,49 kB/s, o rašymo greitis 16,82 ir 64,48 kB/s priklausomai nuo naudojamo dažnio.

Šį realizacija skirta PIC šeimos mikrovaldikliui ir yra Swordfish kompiliatoriaus sudedamoji dalis. Sistemoje ribotas galimas failų skaičius iki 127, todėl kūrėjai siūlo nuspręsti kiek failų bus naudojama, nes valdiklio atmintis yra eikvojama. Testavimo rezultatai pateikti senesnės sistemos versijos, kai buvo palaikoma tik FAT16. Naujausia 4.1.4 versija palaiko FAT32 failinę sistemą ir naujo tipo didelės talpos SDHC kortelės.

Kaip ir ankščiau minėtos sistemos, šioje valdyti valdiklio atliekamas funkcijas yra sukurtas komandų sąrašas (*CloseAll, SaveAll, FreeFile* ir pan.). Realizacijoje galimas jau egzistuojančio failo perrašymas, taip pat subkatalogų kūrimas, trinimas ir keitimas.



20 pav. **SDFileSystem failinės sistemos realizacijos schema PIC18F šeimos mikrovaldikliui [20]**

Pateiktame realizacijos schemoje (žr. 20 pav.) matoma gana nesudėtinga architektūra. Naudojami vos keli elementai, taip pat reikėtų atkreipti dėmesį, jei mikrovaldiklio maitinimo įtampa 5V, būtinai turi būti naudojamas įtampos keitiklis į 3,3V, nes MMC/SD kortelės veikia būtent prie tokios įtampos.

Aptarus failinės sistemos realizacijos skirtos mikrovaldikliui duomenis, galima daryti prielaidą, kad realizacija nepasižymį dideliu duomenų perdavimo greičiu ir specialiomis funkcijomis. Jos naudojimo pasirinkimą lemią PIC šeimos mikrovaldiklių failinių sistemų realizacijų trūkumas. Nes kuriamų sistemų pagrindas sudarytas iš realizacijų skirtų AVR šeimos mikrovaldikliams.

Ketvirta tiriama Analog Devices sukurta failinės sistemos realizacija ir pateikiami jos duomenys.

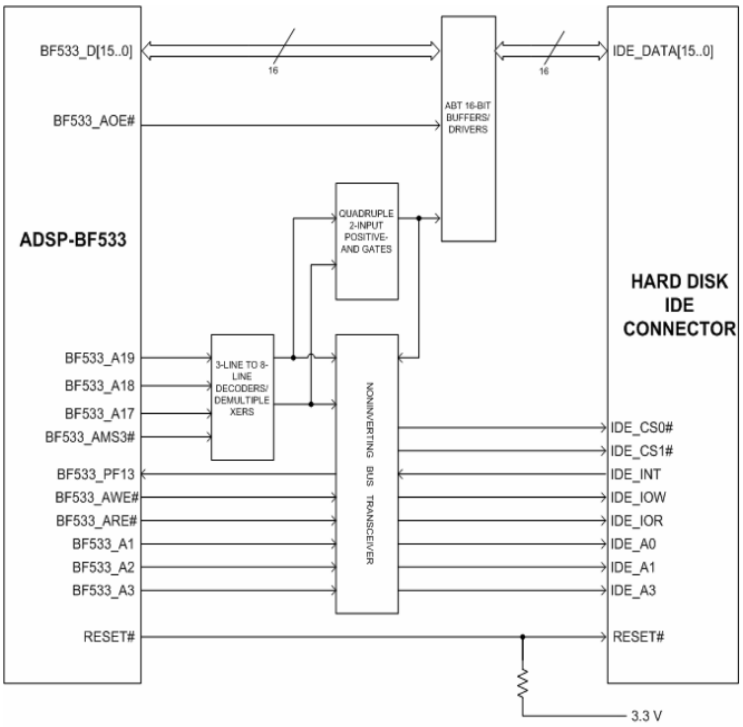
7 lentelė. Failinės sistemos realizacijos skirtos Blackfin ADSP-BF533 skirtos mikrovaldikliui pagrindiniai duomenys

PAVADINIMAS	FAT32 File systems on ADSP-bf533 Blackfin
Kūrėjas	Analog Devices
Realizacijos prieiga per internetą	http://www.analog.com/static/imported-files/application_notes/EE289.ADSP.BF533.Rev1.Feb.2006.pdf
Testavimui naudojama laikmena	Standusis diskas, talpa nežinoma
Realizacijoje naudojamas procesorius	Blackfin ADSP-BF533
Realizacijos veikimui naudojamas dažnis (MHz)	118
Realizacijoje naudojama FAT failinė sistema	FAT32
Pasiektas maksimalus duomenų perdavimo greitis (Įrašymas)	Perdavimo greitis(MB/s)
	6
Pasiektas maksimalus duomenų perdavimo greitis (Skaitymas)	Perdavimo greitis(MB/s)
	8.3

Iš pateiktų duomenų 7 lentelėje galima matyti, kad realizacija skirta Blackfin ADSP-BF533 procesoriui su 188 MHz veikimo dažniu. Testavimui naudotas standusis diskas suformatuotas FAT32 failine sistema ir pasiekti itin aukšti duomenų perdavimo greičiai (skaitymo 8,3 MB/s ir rašymo 6 MB/s).

Realizacija sukurta kompanijos *Analog Devices*, kuri yra gana tvirtai įsitvirtinusi elektronikos rinkoje. Kompanija gamina *Blackfin* mikrovaldikliius su DPS (angl. *Digital Dignal Processing*) funkcija. Šioje realizacijoje naudojamas Blackfin ADSP-BF533 mikrovaldiklis, kuris dažnai aptinkamas įvairiuose mobiliųjų įrenginių (nešiojamuose muzikos ar video grotuvuose ar vaizdo įrašymo įrenginiuose) projektuose. Šio projekto tikslas pritaikyti didelės talpos laikmeną vienam iš ankščiau minėtų įrenginių, kuriam tinkamai funkcionuoti reikalinga didelė laikmenos talpa. Realizacija pritaikyta standžiajam diskui, todėl perdavimo greitis, nepalyginamai didesnis nei SD kortelėse. *Blackfin* mikrovaldiklio darbinis dažnis 118 MHz, kuris penkis ar daugiau kartų didesnis nei naudojamas prieš tai aptartose realizacijose. Visi šie privalumai realizacijai leidžia pasiekti įspūdinga duomenų perdavimo greitį (8,3 MB/s skaitymo ir 6 MB/s įrašymo).

Kūrėjų sprendimas – procesorių prijungti prie standžiojo disko IDE jungties. *ADSP-BF533* asinchroninės atminties prijungimas prie standžiojo disko atliekamas *EBIU* (*External Bus Interface Unit*) pagalba. *EBIU* sąsaja naudoja procesoriaus sisteminį takto laikrodį (*SCLK*).



21 pav. **Blackfin ADSP-BF533 procesoriaus ir standžiojo disko sąsaja [2]**

Schemoje (žr. 15 pav.) matomas procesoriaus (*ADSP-BF533*) ir standžiojo disko (*IDE*) jungties kontaktų sujungimai. Standžiojo disko (*IDE*) jungties kontaktai:

- CS [1:0] naudojamas komandinio registro parinkimui
- A [2:0] 3 bitų ilgio adresas naudojamas pasiekti standžiojo disko duomenų išvadus
- DATA [15:0] Duomenų linijos
- /IOR skaitymo kontaktas
- /IOW įrašymo kontaktas
- /RESET atstatymo kontaktas
- INT funkcijos nutraukimo kontaktas (naudojamas kiekvieno bloko pradžioje)

8 lentelė. Failinės sistemos skirtos ADSP-bf533 Blackfin mikrovaldikliui bandymo rezultatai

Komandos pavadinimas	Ciklų skaičius	Pastabos
Flist	970195	C:/ šakniniame kataloge 12 failų
Fsearch	6542659	Dviejų sluoksnių gylio katalogai
Frename	17959442	
Fopen	961171	Atidaromas 717 MB failas
Fseek	306	Palyginamas 13 kartų
Fclose	149	
fcloselist	2753	

Taigi procesorius pilnai valdo standujį diską aukščiau išvardintų kontaktų pagalba. Aparatūra apjungta programine įranga parašyta C programavimo kalba, kuri turi penkis sluoksnius (Funkcijų biblioteka, FAT sąsajos funkcijos, FAT tvarkyklė, IDE sąsajos funkcijos ir IDE tvarkyklė). Yra sukurtas komandų sąrašas, skirtas valdyti programą. Kaip pavyzdys pateikiami realizacijos bandymo rezultatai (žr. 8 lentelė).

Penkta tiriama Bill Greiman sukurta failinės sistemos realizacija ir pateikiami jos duomenys.

9 lentelė. Failinės sistemos realizacijos skirtos ATmega238P mikrovaldikliui pagrindiniai duomenys

Pavadinimas	SdFatLib			
Kūrėjas	Bill Greiman			
Realizacijos prieiga per internetą	http://code.google.com/p/sdfatlib/			
Testavimui naudojama laikmena	Duomenys nepateikiami		Duomenys nepateikiami	
Realizacijoje naudojamas mikrovaldiklis	ATmega238P (*Arduino* platforma)		ATmega238P (*Arduino* platforma)	
Realizacijos veikimui naudojamas dažnis (MHz)	16		16	
Realizacijoje naudojama FAT failinė sistema	FAT16		FAT32	
Pasiektas maksimalus duomenų perdavimo greitis (Įrašymas)	Failo dydis (B)	Perdavimo greitis(kB/s)	Failo dydis (B)	Perdavimo greitis(kB/s)
	Duomenys nepateikiami	Duomenys nepateikiami	Duomenys nepateikiami	Duomenys nepateikiami
Pasiektas maksimalus duomenų perdavimo greitis (Skaitymas)	Failo dydis (B)	Perdavimo greitis(kB/s)	Failo dydis (B)	Perdavimo greitis(kB/s)
	Duomenys nepateikiami	Duomenys nepateikiami	Duomenys nepateikiami	Duomenys nepateikiami

Iš pateiktos lentelės matyti failinės sistemos realizacija skirta ATmega238P mikrovaldikliu veikiančių 16 MHz taktiniu dažniu, kuris montuojamas *Arduino* platformoje. Sistema palaiko dvi failinės sistemas FAT16 ir FAT32. Apie duomenų perdavimo greitį duomenų internetinėje svetainėje nepateikiama. Susisiekus su kūrėju elektroniniu paštu, bandyta gauti daugiau informacijos, tačiau konkretaus atsakymo negauta.

Sistemos kodas parašytas gimininga C kalba, palaiko failų sukūrimą, ištrinimą, rašymą, skaitymą ir pervadinimą, taip pat yra galimybė kurti ir naikinti aplankus, bei sub-aplankus. SdFatLib palaiko tik trumpus 8.3 failų vardus. Galimos naudoti SD, microSD, SDHC atminties kortelės.

Taigi ištyrus penkias failines sistemas skirtas mikrovaldikliams ir įvertinus visas galimybes, pasirinkta penktoji realizacija, dėl savo bandymo ir duomenų perdavimo greičio informacijos stokos bei populiaraus ir pakankamai nebrangaus mikrovaldiklio (ATmega328P) naudojimo lanksčioje platformoje. Kitos failinių sistemų realizacijos atmestos dėl brangių ir didelių gabaritų standžiųjų diskų ir nepopuliarių mikrovaldiklių naudojimo. Sekančiame skyriuje ši sistema bus tiriama duomenų rašymo, skaitymo ir mikrovaldiklio taktinio dažnio keitimo atžvilgiu.

3. FAILINĖS SISTEMOS REALIZACIJOS SKIRTOS MIKROVALDIKLIAMS TOBULINIMAS

Šiame skyriuje bus aptariamas: failinės sistemos skirtos mikrovaldikliui technologijų pasirinkimas, kokie kriterijai jį lėmė, pasirinktų technologijų pagrindiniai privalumai; failinės sistemos realizacijos skirtos mikrovaldikliui tobulinimas panaudojant pasirinktas technologijas taikant literatūros turinio analizės surinktus duomenis. Tyrimu siekiama nustatyti, kokią įtaka failinės sistemos realizacijos duomenų perdavimo greitaveikai daro: mikrovaldiklio veikimo dažnis, failinė sistema, siunčiamų duomenų kiekis, SD kortelė ir duomenų perdavimo sąsaja. Pateikiami tyrimo metu gauti sintezavimo ir modeliavimo duomenys, apibendrinami gauti rezultatai.

3.1. Pasiruošimas failinės sistemos realizacijos skirtos mikrovaldikliui tobulinimui

Norint atlikti failinės sistemos realizacijos skirtos mikrovaldikliui tyrimą ir tobulinimą, būtina pasirinkti tarpusavyje susiejamas technologijas, galinčias duoti geriausią rezultatą, todėl šiame skyriuje nagrinėjamos aktualiausios technologijos. Pateikiama pasirinkta failinės sistemos realizacija skirta mikrovaldikliui, aprašomos jos pagrindinės funkcijos, lėmusios pasirinkimą; pasirinktas mikrovaldiklis ir jo veikimui skirta platforma; elektroninė laikmena, kurioje saugomi duomenys ir suformatuota pasirinkta failine sistema; duomenų perdavimo sąsaja, kurios pagalba duomenys perduodami iš mikrovaldiklio į SD kortelę; programuojamoji matrica, kuri padės suprogramuotam mikrovaldikliui išvysti didesnį veikimo dažnį; aparatūros projektavimo kalba, kuria bus aprašomas mikrovaldiklis ir jo registrai.

FatSdLib

Išsamesniam tyrimui ir tobulinimui pasirinkta Bill Greiman FatSdLib failinės sistemos realizacija skirta *Arduino* platformai su Atmel kompanijos ATMEGA328P mikrovaldikliu. Failinės sistemos realizacijos pasirinkimą lėmė:

- bandymo ir duomenų perdavimo greičio informacijos stoka;
- pritaikymas populiariai *Arduino* platformai;
- atviras ir laisvai prieinamas realizacijos kodas;
- FAT16 ir FAT32 failinių sistemų palaikomumas;
- padidintos talpos SDHC kortelių palaikomumas;
- aparatūrinės ir programinės SPI sąsajos palaikomumas;
- programavimo kalba gimininga C kalbai;
- specialių simbolių palaikomumas.

Arduino

Pirmas tyrimo etapas atliekamas naudojant *Arduino* platformą - atviro kodo, lanksti elektronikos prototipų kūrimo sistema. Šios technologijos pasirinkimą lėmė:

- sistema pritaikyta patogiam varotojo naudojimui, patogiai išdėstyti išvesties/įvesties išvadai;
- įdiegtas RS232 sąsajos konverteris į USB (kas šiuo metu labai populiaru);
- optimalus įvesties (7-12 V) ir išvesties (3,3 arba 5 V) įtampos diapazonas;
- mikrovaldiklio (šiuo atveju ATmega238P) atminties talpa - 32 KB;
- veikimo taktinis dažnis 16 MHz;
- populiarumas, kurio pagalba galima rasti daugiau informacijos internetinėje erdvėje;
- Programavimo kalba gimininga C kalbai.

SD kortelė

Failinės sistemos realizacijos skirtos mikrovaldikliui tyrime ir tobulinime bus naudojama SD kortelė. Pasirinkimą lėmė:

- failinės sistemos palaikomumas (šiuo atveju FAT16 ir FAT32);
- laikmenos dydis (32mm ilgis, 24mm plotis ir 2.1mm storis);
- duomenų perdavimo sparta;
- didelė talpa;
- nedidelė kaina.

SPI

Šiuolaikiniuose mikrovaldikliuose yra realizuotos kelios duomenų perdavimo sąsajos. Tyrime naudojama SPI duomenų perdavimo sąsaja, dėl jos populiarumo duomenų perdavime SD kortelėse naudojant mikrovaldiklius. Šią sąsaja galima naudoti įgyvendinta programiškai arba aparatūriškai. SPI veikimui reikalingi keturi kontaktai CS (*chip select*), SCLK (*clock*), MISO (*master in slave out*), MOSI (*master out slave in*).

FPGA

FPGA pasirinkimas failinės sistemos realizacijos tobulinimui buvo apgalvotas nuo pat darbo pradžios, nes šios technologijos pranašumas prieš daugumą elektrinių grandinių yra lankstumas ir universalumas grįstas išskirtine architektūra. Tai - puslaidininkių elementų (pvz., trigerių, multiplekserių ar sumatorių) laukas, tarpusavyje sujungiamų perprogramuojamais loginiais ryšiais. Šios savybės išskiria FPGA iš kitų elektroninių prietaisų kaip sistemą, leidžiančią įgyvendinti įvairaus tipo uždavinius, sukuriant kompleksinės logines grandines.

Loginėms sąsajoms tarp vidinių FPGA blokų sudaryti bei grandinėms įgyvendinti yra skirtos techninio lygmens programavimo kalbos VHDL ir Verilog. Perprogramuojama logika leidžia projektuotojui nebrangiai modeliuoti schemas įgyvendinant jas viena nedidele sistema. FPGA yra universalus sprendimas moksliniams tyrimams.

Tyrimui ir eksperimentams naudota FPGA – Xilinx Spartan-3AN Starter Kit XC3S7000AN, programinė įranga Xilinx ISE Design Suite web pack 13.1. ISE Design Suite susideda iš keleto aplikacijų, pagrindinės yra:

- ISE (pagrindinio projekto kūrimo įrankis);
- EDK (sistemos su valdikliu (MicroBlaze, PowerPC) ir visapusiška periferija kūrimo įrankis);
- SDK (programinės dalies (C programavimo kalba), skirtos valdikliui kūrimo įrankis).

VHDL

VHDL – skaitmeninės elektroninės aparatūros funkcionavimo aprašymo kalba - laikoma standartu. VHDL failinės sistemos realizacijos aparatūros aprašymui ir programavimui pasirinkta dėl jos pritaikymo grandinių sintezei ir modeliavimui. Pagrindinis motyvuojantis VHDL aspektas - jos standartiška ir technologiškai nepriklausoma kalba, kuri yra portatyvi ir daugkartinio naudojimo.

3.2. Failinės sistemos realizacijos skirtos mikrovaldikliui tobulinimo eiga

Pasirinkus failinės sistemos realizaciją skirtą mikrovaldikliui, naudojamą aparatūrą, perdavimo sąsają ir tobulinimui reikalingus įrankius aptariama darbo eiga. Aptariami įrankiai naudojami pasirinktos failinės sistemos realizacijos tyrime, programavimo kodai bei jų paaiškinimai. Toliau nagrinėjama, kaip tyrėjas tobulina pasirinktą ir ištestuotą realizaciją, kokie sprendimai priimti siekiant geriausio rezultato.

Kaip minėta ankščiau pasirinktos failinės sistemos realizacija aprašyta C kalbos gimiringu kodu (Wiring), todėl svarbu suprogramuoti reikiamą programos kodą, kuris generuoja programą atliekančią planuotus veiksmus. Žemiau pateikiamas programos kodas, kuris leidžia nustatyti duomenų perdavimo (rašymo/skaitymo) greitį pasirinktomis sąlygomis.

```
#include <SdFat.h>
#include <SdFatUtil.h>
#define FILE_SIZE_MB 5
#define FILE_SIZE (1000000UL*FILE_SIZE_MB)
#define BUF_SIZE 100

uint8_t buf[BUF_SIZE];
Sd2Card card;
SdVolume volume;
```

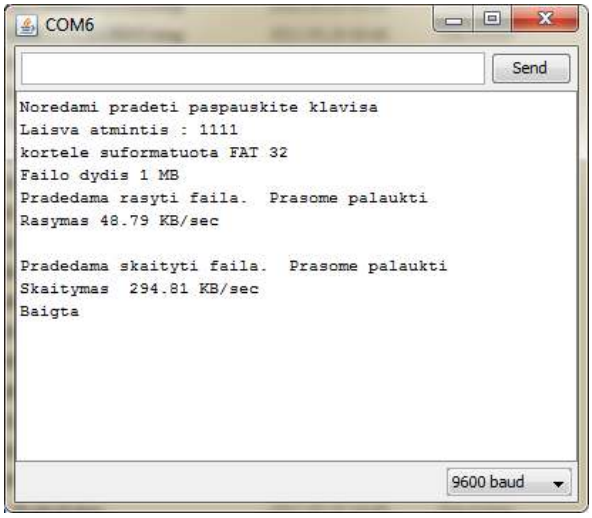
```
SdFile root;
SdFile file;
#define error(s) error_P(PSTR(s))
void error_P(const char* str)
{
  PgmPrint("klaida: ");
  SerialPrintln_P(str);
  if (card.errorCode()) {
    PgmPrint("SD kortelės klaida: ");
    Serial.print(card.errorCode(), HEX);
    Serial.print(',');
    Serial.println(card.errorData(), HEX);
  }
  while(1);
}

void setup() {

  Serial.begin(9600);
  PgmPrintln("Noredami pradeti paspauskite klavisa");
  while (!Serial.available());

  PgmPrint("Laisva atmintis : ");
  Serial.println(FreeRam());

  if (!card.init(SPI_FULL_SPEED,AL4)) error("korteles klaida");

  if (!volume.init(&card)) error("FAT klaida");

  PgmPrint("kortele suformatuota FAT ");
  Serial.println(volume.fatType(), DEC);

  if (!root.openRoot(&volume)) error("Klaida");

  if (!file.open(&root, "BENCH.DAT", O_CREAT | O_TRUNC | O_RDWR)) {
    error("Klaida atidarant faila");
  }
  for (uint16_t i = 0; i < (BUF_SIZE-2); i++) {
    buf[i] = 'A' + (i % 26);
  }
  buf[BUF_SIZE-2] = '\r';
  buf[BUF_SIZE-1] = '\n';

  PgmPrint("Failo dydis ");
  Serial.print(FILE_SIZE_MB);
  PgmPrintln(" MB");
  PgmPrintln("Pradedama rasyti faila.  Prasome palaukti");

  uint32_t n = FILE_SIZE/sizeof(buf);
  uint32_t t = millis();
  for (uint32_t i = 0; i < n; i++) {
    if (file.write(buf, sizeof(buf)) != sizeof(buf)) {
      error("Rasymo klaida");
    }
  }
  t = millis() - t;
  file.sync();
  double r = (double)file.fileSize()/t;
  PgmPrint("Rasymas ");
  Serial.print(r);
  PgmPrintln(" KB/sec");
  Serial.println();
  PgmPrintln("Pradedama skaityti faila.  Prasome palaukti");
    file.rewind();
  t = millis();
  for (uint32_t i = 0; i < n; i++) {
    if (file.read(buf, sizeof(buf)) != sizeof(buf)) {
      error("Skaitymo klaida");
    }
  }
  t = millis() - t;
  r = (double)file.fileSize()/t;
  PgmPrint("Skaitymas  ");
  Serial.print(r);
  PgmPrintln(" KB/sec");
  PgmPrintln("Baigta");
}
void loop() { }
```

Kodas susideda iš dviejų pagrindinių funkcijų *setup()* ir *loop()*. *Setup* funkcijoje aprašomi kintamieji, kontaktų reikšmės, nurodomos bibliotekos, *loop* funkcijoje aprašomi programos veiksmai, kurie valdo sistemą. Šiuo atveju yra naudojamos bibliotekos, kuriose jau aprašytos pagrindinės veikimo funkcijos, todėl naudojama tik *setup* funkcija. Pirmos dvi kodo eilutės nurodo įtraukiamas bibliotekas, *SdFat.h* – failinės sistemos ir *SdFatUtil.h* – papildomų funkcijų biblioteka. Sekančios trys kodo eilutės nurodo failo ir naudojamo buferio dydi. Programoje realizuota klaidų aptikimo sistema, kuri tikrina ar SD kortelė tinkamai prijungta, ar jos failinė sistema atitinka reikalavimus ir ar testinis failas sėkmingai skaitomas/rašomas. Sekančiu veiksmu programa nustato RS232 sąsajos spartą bodais (angl. *baud rate*) ir prašo varotojo pradėti testavimą klavišo paspaudimu, vartotojui atlikus šia funkciją ir paleidus testavimą, ekrane parodoma laisva atmintis mikrovaldiklyje (iš papildomų funkcijų bibliotekos), SD kortelės failinės sistemos pavadinimas, pasirinktas testavimo failo dydis ir pradedama kortelės inicializacija naudojant SPI sąsają. Jei inicializacija nesėkminga komandiniame lange atvaizduojamas klaidos kodas, jei veiksmas atliktas be klaidų, programinis buferis užpildomas duomenimis bei kuriamas testinis failas (TEST.DAT). Sekantys veiksmai atlieka sukurto failo rašymą ir skaitymą į/iš SD kortelės ir pateikiami rezultatai kilobaitų per sekundę (KB/s) tikslumu.

22 pav. ***Arduino* programavimo aplinka**

Kodas rašomas į *Arduino* programavimo aplinką (žr. 17 pav.). Pasirenkamas naudojamas mikrovaldiklis (šiuo atveju ATmega328P) ir kodas kompiliuojamas, jei sistema neaptinka

programavimo klaidų, kodą galima nusiųsti mikrovaldikliui. Kodas parašytas C ar panašia programavimo kalba paprastai nėra palaikomas mikrovaldiklių, todėl programinėje įrangoje, kuri komunikuoja su mikrovaldikliais, programos kodas yra konvertuojamas į .HEX failą. *Arduino* programinė įranga ne išimtis, todėl paspaudus įkėlimo (angl. *upload)* mygtuką, atktyvuojamas gcc kompiliatorius ir sukuriamas .HEX failas (Windows 7 operacinėje sistemoje jis dažniausiai konvertuojamas C:\Users\varotojas\AppData\Local\Temp\buildxxx aplanke) ir įkeliamas į mikrovaldiklio atmintį. Sėkmingai atlikus veiksmus, galima pradėti stebėti programos veikimą aparatūroje, tai atliekama RS232 į USB sąsajos konverterio pagalba, kuris sukuria virtualų išvadą (šiuo atveju COM6) ir naudojant komandinį langą gaunama informacija.



23 pav. **Failinės sistemos realizacijos testavimo gauti rezultatai**

Kaip galima matyti iš gautų rezultatų (žr. 18 pav.) programa failinės sistemos realizacijai veikia kaip tikėtasi, todėl galima atlikti išsamesnį jos tyrimą.

Tyrimui pasirinktos trijų skirtingų gamintojų SD kortelės (Lexar 512MB, Apacer 2 GB) ir microSD kortelė (Kingston 1 GB). Visos kortelės testuojamos tokiomis pačiomis sąlygomis, naudojant tą patį mikrovaldiklį, programinę įrangą, programinę ir aparatūrinę SPI sąsają. Svarbu sudaryti vienodas sąlygas, norint pasiekti tikslius rezultatus, išvengti įvairių nukrypimų nuo normos ir trikdžių. Žemiau pateikiamos lentelės su rezultatais *Arduino* platformoje su ATmega328P mikrovaldikliu atitinkamai FAT16 ir FAT32 failinėse sistemose.

10 lentelė. SD kortelių duomenų perdavimo greitis FAT16 failinėje sistemoje naudojant aparatūrinę SPI sąsają

FAT16	Lexar 512 MB		Apacer 2 GB		microSD Kingston 1 GB	
Failo dydis (MB)	Rašymo greitis (KB/s)	Skaitymo greitis (KB/s)	Rašymo greitis (KB/s)	Skaitymo greitis (KB/s)	Rašymo greitis (KB/s)	Skaitymo greitis (KB/s)
1	273,43	306,68	96,64	271,52	159,21	331,56
5	276,34	306,9	113,32	271,37	166,79	331,90
10	276,86	306,52	109,78	269,19	165,35	331,48

Iš gautų rezultatų naudojant failinę sistema FAT16 SD kortelėje matyti (žr. 9 lentelė), kad kortelių skaitymo ir rašymo greitis nuo failo dydžio kinta nežymiai (svarbu paminėti, kad rezultatai atliekant testavimą gali kisti dėl vartotojui nepastebimų įtampos šuolių, sistemos sujungimo įrangos ir pan.)

Taigi gauti beveik vienodi perdavimo greičiai, iš to galima daryti prielaidą, kad įtakos greičiui turi kortelės gamintojas, talpa, susidėvėjimo faktorius ir kt.

11 lentelė. SD kortelių duomenų perdavimo greitis FAT32 failinėje sistemoje naudojant aparatūrinę SPI sąsają

FAT32	Lexar 512 MB		Apacer 2 GB		microSD Kingston 1 GB	
Failo dydis (MB)	Rašymo greitis (KB/s)	Skaitymo greitis (KB/s)	Rašymo greitis (KB/s)	Skaitymo greitis (KB/s)	Rašymo greitis (KB/s)	Skaitymo greitis (KB/s)
1	79.21	278.61	106.33	293.74	24,61	312,11
5	93.14	289.49	106.6	293.76	22,81	311,97
10	112.58	290.51	106.39	293.25	19,64	331,48

Iš gautų rezultatų naudojant failinę sistema FAT32 matyti, kad naudojant didesnius failus Laxar SD kortelės duomenų rašymo greitis tendencingai kyla. Kingston microSD kortelėje pastebimas atvirkščias reiškinys, kuo didesnis failo dydis – tuo mažesnis rašymo greitis. Apacer kortelėje rašymo greitis nepriklausomas nuo failo dydžio. Skaitymo greitis visose kortelėse kinta nežymiai.

Taigi gauti rašymo į SD kortelę duomenų perdavimo greičiai mažesni nei naudojant FAT16 failinę sistemą, galima teigti, kad įtaką daro failinės sistemos paskirstymo vieneto dydis (allocation unit size). Naudojant FAT16 failinę sistemą paskirstymo vieneto dydis buvo – 16384 baitų, FAT32 – 4096 baitų.

Norint išsamiau nagrinėti duomenų perdavimo greitį SD kortelėse, tyrimui naudojama programinė SPI sąsaja. Atliekami kodo pakeitimai failinės sistemos realizacijos bibliotekoje (Sd2Card.h), kurie įjungia programinę ir atjungia aparatūrinę SPI sąsają:

```
#define MEGA_SOFT_SPI 1
#define SOFTWARE_SPI

#else uint8_t spiRec(void) {
  uint8_t data = 0;
  cli();
  fastDigitalWrite(SPI_MOSI_PIN, HIGH);

  for (uint8_t i = 0; i < 8; i++) {
    fastDigitalWrite(SPI_SCK_PIN, HIGH);

    nop;
    nop;
    data <<= 1;
    if (fastDigitalRead(SPI_MISO_PIN)) data |= 1;
    fastDigitalWrite(SPI_SCK_PIN, LOW);
  }
  sei();
  return data;
}
void spiSend(uint8_t data) {
  cli();
  for (uint8_t i = 0; i < 8; i++) {
    fastDigitalWrite(SPI_SCK_PIN, LOW);
    fastDigitalWrite(SPI_MOSI_PIN, data & 0X80);
```

```
    data <<= 1;
    fastDigitalWrite(SPI_SCK_PIN, HIGH);
  }
  nop;
  nop;
  nop;
  nop;
  fastDigitalWrite(SPI_SCK_PIN, LOW);
  sei();
}
#endif
```

Pirmos dvi kodo eilutės aktyvuoja programinę SPI sąsają, o sekančios eilutės aprašo ją programiškai, matomos aprašytos duomenų siuntimo ir gavimo funkcijos. Taigi atlikus šiuos pakeitimus galima atlikti SD kortelių testavimą, naudojant ta pačią programinę bei aparatūrinę įrangą.

12 lentelė. SD kortelių duomenų perdavimo greitis FAT16 failinėje sistemoje naudojant programinę SPI sąsają

FAT16	Lexar 512 MB		Apacer 2 GB		microSD Kingston 1 GB	
Failo dydis (MB)	Rašymo greitis (KB/s)	Skaitymo greitis (KB/s)	Rašymo greitis (KB/s)	Skaitymo greitis (KB/s)	Rašymo greitis (KB/s)	Skaitymo greitis (KB/s)
1	81,12	102,61	47,15	98,74	66,32	103,99
5	81,66	102,77	52,50	100,35	67,63	104,04
10	81,68	102,72	58,37	100,52	67,80	103,99

Matomas žymus duomenų perdavimo greičio sumažėjimas, lyginant su aparatūrinę SPI sąsaja, galima daryti prielaidą, kad tam įtakos turi veikimo dažnis. Mikrovaldiklio viduje esanti SPI veikia didesniu dažniu nei programinė SPI sąsaja.

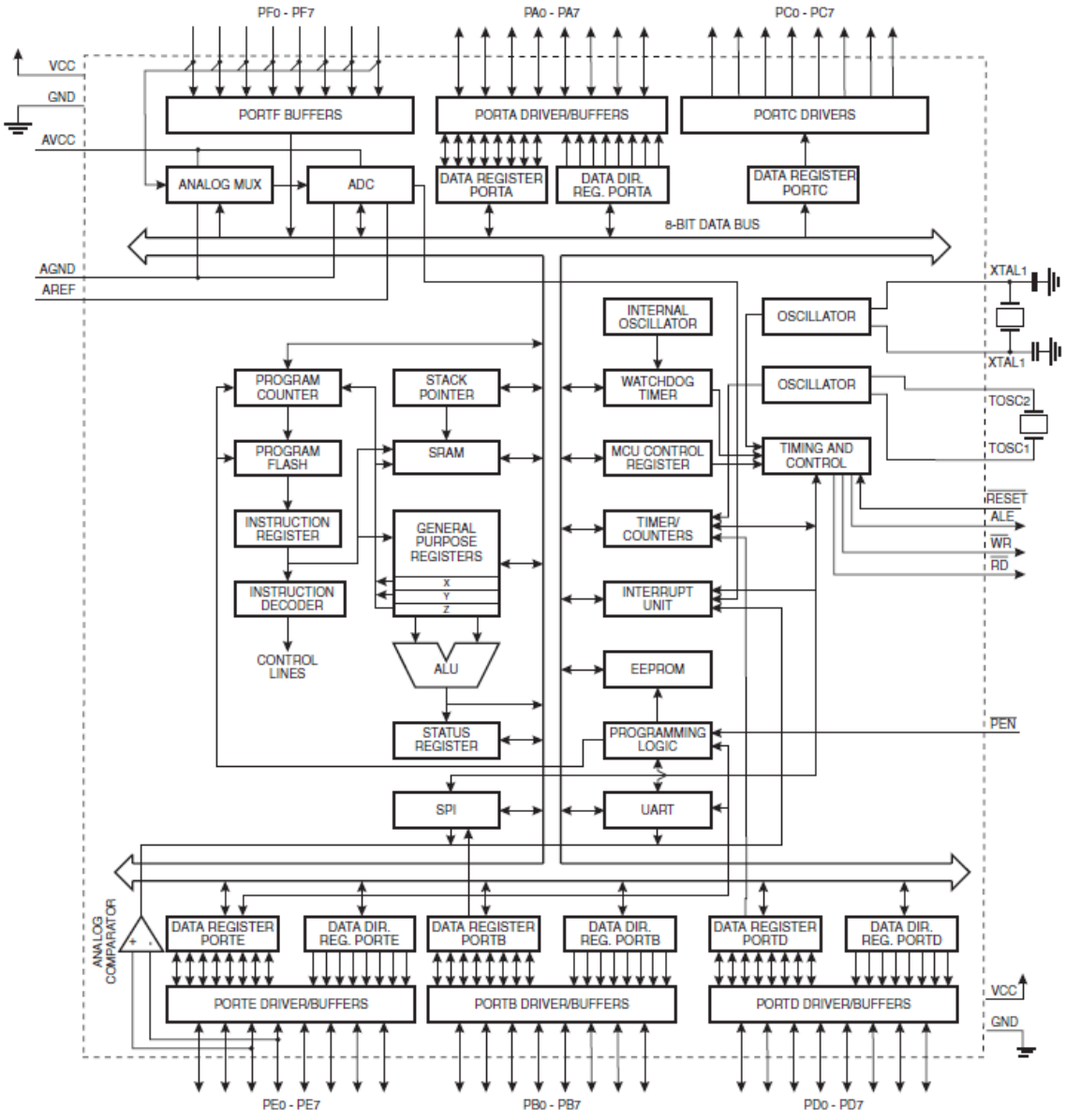
13 lentelė. SD kortelių duomenų perdavimo greitis FAT32 failinėje sistemoje naudojant programinę SPI sąsają

FAT32	Lexar 512 MB		Apacer 2 GB		microSD Kingston 1 GB	
Failo dydis (MB)	Rašymo greitis (KB/s)	Skaitymo greitis (KB/s)	Rašymo greitis (KB/s)	Skaitymo greitis (KB/s)	Rašymo greitis (KB/s)	Skaitymo greitis (KB/s)
1	31,90	94,80	49,73	91,60	20,23	96,08
5	31,91	94,83	51,30	91,91	19,08	96,11
10	31,88	94,78	51,89	91,72	16,83	96,06

Apibendrinant galima teigti, kad duomenų įrašymui į SD kortelę didelę įtaką turi failinės sistemos pasirinkimas bei naudojamas paskirstymo vieneto dydis. SD kortelės failų skaitymo greitis beveik nekinta naudojat skirtingas failines sistemas. Taip pat labai didelis greičių skirtumas tarp aparatūrinės ir programinės SPI duomenų perdavimo sąsajos, todėl projektuojant įrenginius kur reikalinga didesnė duomenų perdavimo sparta, patartina naudoti aparatūrinę SPI sąsają.

Toliau nagrinėjamas pasirinktos Bill Greiman failinės sistemos realizacijos skirtos mikrovaldikliui tobulinamas, pasirinktos technologijos ir įrankiai teigiamam rezultatui pasiekti.

Mikrovaldiklis AVR aprašomas VHDL aparatūros kalba, tai atliekama tam, kad būtų galima modeliuoti grandinę prieš atliekant sintezę ir įkėlimą į programuojamą matricą. Todėl būtina išsiaiškinti mikrovaldiklio sudedamuosius elementus, žemiau pateikiama AVR mikrovaldiklio blokine diagrama (žr. 24 pav.).

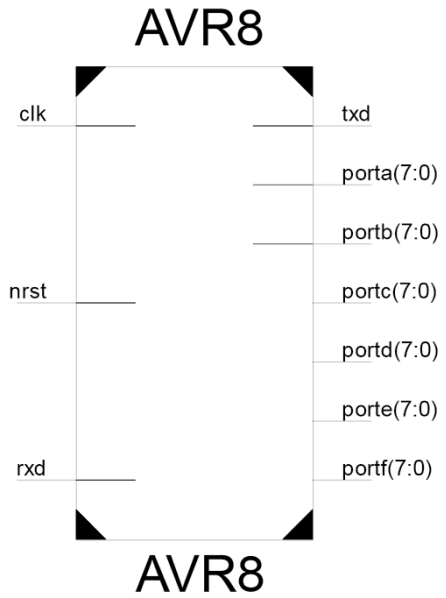


24 pav. **AVR 8 bitų mikrovaldiklio blokinė diagrama**

Iš diagramos matyti, kad mikrovaldiklis sudarytas iš: šešių 8 bitų skirtingų išvadų (PA0 – PA7, PB0 – PB7, PC0 – PC7, PD0 – PD7, PE0 – PE7, PF0 – PF7); atminties SRAM, EEPROM, program flash; būsenos registrų; aritmetinių-loginių įrenginių (ALU); laikmačio (timer) ir pan. Todėl dauguma šių elementų turi būti aprašyti VHDL kodu, kad veiksmai atliekami naujuoju

programiniu mikrovaldikliu atitiktų etaloną. Šiam etapui naudojamas kodas parašytas rusų projektuotojo Ruslan Lepetenok - sukurtas programinis AVR mikrovaldiklis.

Pakoregavus VHDL kodą pagal poreikius, t.y. prijungus programos atmintį prie mikrovaldiklio ir atjungus išorinius elementus, mikrovaldiklis sintezuojamas ir gaunamas elemento vaizdas (žr 25 pav.) Visa grandinė pateikiama priede nr. 1. Sintezavimui naudojama Xilinx kompanijos programinė įranga ISE 13.1 versija.

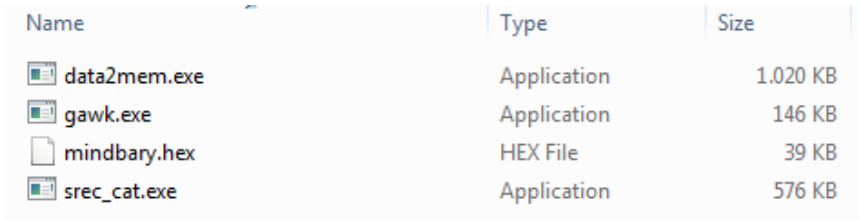


25 pav. **Sintezuotas AVR procesoriaus vaizdas**

Taigi po sintezės gautų teigiamų rezultatų, buvo sukurtas .BIT failas, kuriame saugoma visa sintezės metu sukaupta informacija. Šį failą reikia naudoti įkraunant i FPGA programuojamąją matricą, bei konvertuojant failus.

Norint atlikti testavimą failinės sistemos realizacijai skirtai parašytam VHDL aparatūros kalba mikrovaldikliui, būtina programos kodą (žr. 22 pav.) konvertuoti į VHDL failą ir prijungti prie naujo mikrovaldiklio, kaip programos atmintį. Kaip jau minėta ankščiau *Arduino* programinė įranga sukuria .HEX, kurį įkelia į mikrovaldiklį, taigi sekančiais žingsniais atptariama, kaip konvertuotą programos kodą .HEX faile konvertuoti į VHDL kodą.

Atlikti aukščiau minėtus veiksmus būtini trys avr-gcc konverteriai: *data2mem.exe, gawk.exe, srec_cat.exe*.



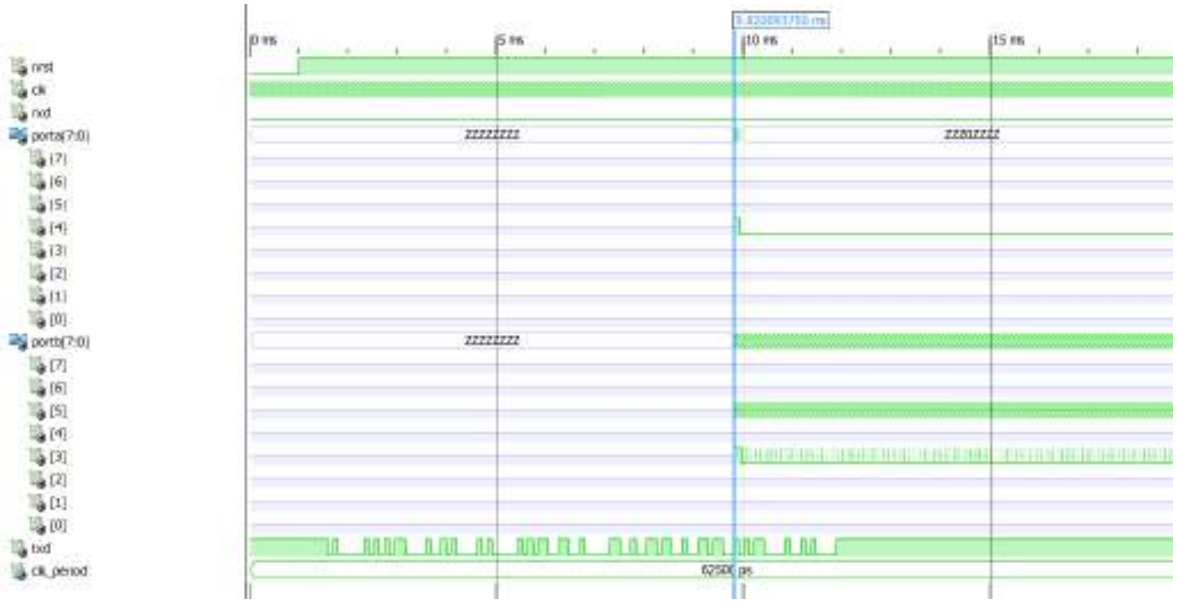
26 pav. **Įrankiai reikalingi konveruoti .HEX failą į VHDL kodą**

Ir atliekamos šie veiksmai komandiniame lange:

```
srec_cat mindbary.hex -Intel -Byte_Swap 2  -Data_Only -o tmp.mem -vmem 8
gawk ' BEGIN{FS=" ";} { $1= ""; print}' tmp.mem > out.mem

data2mem -bm bitstreams/avr8_bd.bmm -bt bitstreams/avr8.bit -bd out.mem -o b out.bit
data2mem -bm bitstreams/avr8_bd.bmm -bt out.bit -d > out.dmp
data2mem -bm bitstreams/avr8_bd.bmm -bd out.mem -o h
```

Čia .bit failas gautas sintezės metu. Po šių veiksmų gaunamas VHDL failas pavadinimu „*prog_mem_init.vhd*“, kuris atitinka naudota programą *Arduino* programinėje įrangoje. Šis failas prijungiamas prie programinio mikrovaldiklio kaip programinė atmintis. Todėl galima parašyti testinį kodą (VHDL kalba) ir patikrinti ar grandinė veikia kaip tikimasi.

Testavimo faile taktinio dažnio periodas nustatomas 62,5 ns atitinkantis 16 MHz dažnį, kuris naudojamas fiziniame AVR mikrovaldiklyje. Sumodeliavus grandinę, gaunami modeliavimo rezultatai (žr. 27 pav.)

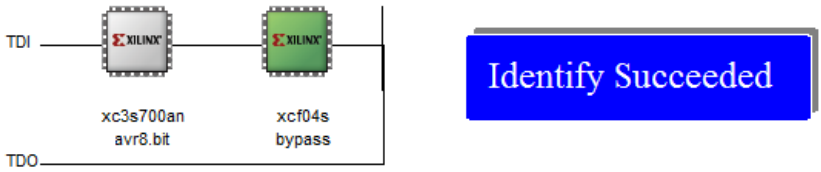


27 pav. **AVR programinio mikrovaldiklio modeliavimo rezultatai**

Iš modeliavimo rezultatų matyti, kortelės inicializacija užtruko 9,8 ms. *Txd* kontaktas apie 12ms transliavo tekstą, kurį vartotojas mato komandiniame lange. Aktyvūs yra 3 kontaktai, *porta* – 4 yra SPI sąsajos SS kontaktas, *portb* – 3 MOSI kontaktas ir *portb* - 5 SCK kontaktas. MISO kontakto nėra todėl, kad joks išorinis įrenginys (SPI *slave*) nepriima duomenų ir jų negrąžina į (SPI *master*), dėl šios priežasties rxd laiko loginį 0. Tačiau tai rodo, kad failinė realizacija siunčia duomenis ir naudojant FPGA lustą prijungtus SD kortelę, programa funkcionuos kaip tikimasi. Todėl galima atlikti duomenų įkėlimą į programuojamąją matricą.

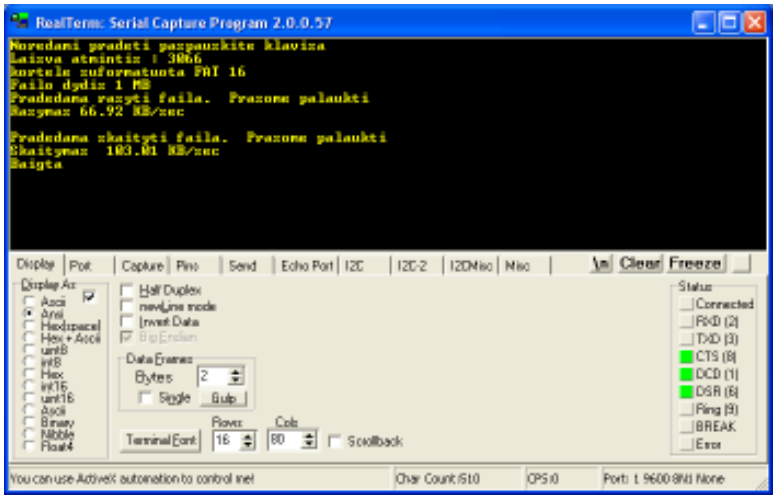
Šiam procesui atlikti būtina darbui paruošti FPGA (įjungti maitinimą, duomenų kabelį, aptikrinti ir ar teisingai sujungti trumpikliai) ir sintezuotą elektrinę grandinę .BIT formatu įkelti į programuojamąją matricą (šiuo atveju Xilinx ISE 13.1 programinio paketo „*configure target devize*“ funkcijos pagalba).

Būtina atkreipti dėmesį, jog sintezuotą grandinę reikia įkelti į pagrindinį FPGA lustą (šiuo atveju xc3700an), nes kitu atveju galima sunaikinti esamą konfigūraciją, saugomą xcf04s luste. Sėkmingai atlikus failo įkėlimą į FPGA, programos ekrane gaunamas tai patvirtinantis pranešimas (žr. 28 pav.).



28 pav. **Sintezuoto failo įkėlimas į FPGA lustą**

Atlikus visus veiksmus be klaidų, gaunamas rezultatas lygiai toks pat kaip ir *Arduino* platformoje su AVR mikrovaldikliu (naudojant taktinį dažnį 16 HMz). Rezultatai siunčiami RS232 sąsaja, todėl kompiuteryje būtinas šios sąsajos išvadas, taip pat galima naudoti išorinį RS232 konverterį. Rezultatų atvaizdavimui ekrane galima naudoti, bet kokį komandinį terminalą (šiuo atveju *RealTerm* programa žr. 29 pav.),



29 pav. **Rezultatų langas naudojant FPGA lustą su integruotu AVR mikrovaldikliu**

Svarbu paminėti, kad FPGA luste naudojama programinė SPI sąsaja, nes programiniame AVR mikrovaldiklyje aparatūrinė SPI sąsaja nerealizuota.

Pagrindinis darbo tikslas - patobulinti duomenų perdavimo greitį, tai galima atlikti didinant FPGA programuojamosios matricos taktinį dažnį, kiek leidžia realizuotas mikrovaldiklis. Tai atliekama DCM (angl. *Digital Clock Manager*) komponento pagalba; įėjimo taktinį dažnį galima keisti į norimą išėjimo dažnį (šį komponentą, būtina įtraukti į projekto mikrovaldiklio kodą ir priskirti naudojamą taktinį laikrodį).

Tyrimui naudojamas FPGA programuojamosios matricos (Xilinx Spartan3AN) komplekte esantis microSD adapteris, todėl tolimesnis tyrimas vyks tik su šia atminties kortele. Taigi keičiant programinio mikrovaldiklio taktinį dažnį tiriama, kaip kinta failinės sistemos

realizacijos duomenų perdavimo greitis, skirtingose sistemose ir siunčiant skirtingus failų dydžius.

14 lentelė. Failinės sistemos realizacijos FPGA luste duomenų perdavimo greičio rezultatai naudojant FAT16 ir skirtingus taktinius dažnius

FAT16	microSD Kingston 1 GB		
Failo dydis (MB)	Taktinis dažnis	Rašymo greitis (KB/s)	Skaitymo greitis (KB/s)
1	32 MHz	130,52	207,57
5		131,15	205,34
10		130,89	205,98
Failo dydis (MB)	Taktinis dažnis	Rašymo greitis (KB/s)	Skaitymo greitis (KB/s)
1	48 MHz	198,05	308,99
5		200,65	306,51
10		203,1	306,73
Failo dydis (MB)	Taktinis dažnis	Rašymo greitis (KB/s)	Skaitymo greitis (KB/s)
1	64 MHz	261,28	414,02
5		262,52	415,87
10		264,02	415,54

Atlikus skirtingų taktinių mikrovaldiklio dažnių testavimą (32, 48 ir 64 MHz) matoma, kad failinės sistemos duomenų perdavimo greitis didėja kylant dažniui. Maksimalus pasiektas greitis 264,02 KB/s rašymo ir 415,87 skaitymo, naudojant 64 MHz taktinį dažnį.

Toliau tyrimas atliekamas FAT32 failinėje sistemoje tokiomis pačiomis sąlygomis.

15 lentelė. Failinės sistemos realizacijos FPGA luste duomenų perdavimo greičio rezultatai naudojant FAT32 ir skirtingus taktinius dažnius

FAT32	microSD Kingston 1 GB		
Failo dydis (MB)	Taktinis dažnis	Rašymo greitis (KB/s)	Skaitymo greitis (KB/s)
1	32 MHz	40,02	193,29
5		39,35	192,58
10		41,86	190,81
Failo dydis (MB)	Taktinis dažnis	Rašymo greitis (KB/s)	Skaitymo greitis (KB/s)
1	48 MHz	61,08	287,69
5		58,98	288,01
10		57,65	289,33
Failo dydis (MB)	Taktinis dažnis	Rašymo greitis (KB/s)	Skaitymo greitis (KB/s)
1	64 MHz	80,92	370,66
5		79,83	372,87
10		77,26	372,91

Taigi atlikus giluminį failinės sistemos realizacijos testavimą ir įvertinus rezultatus matyti, kad greitis tiesiogiai priklausomas nuo taktinio mikrovaldiklio dažnio, t.y. tolygiai didėja keliant dažnį. Taip pat pastebėta, jog naudojant programiškai realizuotą SPI sąsająfailo dydis reikšmingos įtakos neturi, naudojant FPGA ir microSD atminties koretelę (testuojant iki 10 MB failus).

IŠVADOS

- Mokslinės literatūros analizė atskleidė failinių sistemų realizacijų skirtų mikrovaldikliams teorinius aspektus: failinė sistema yra pagrindinė struktūra, kurią naudoja kompiuteris tvarkyti duomenis. Šiuo metu populiariausios failų sistemos, yra: NTFS, FAT16, FAT32. Dvi pagrindinės failinių sistemų taikymo aplinkos yra standieji diskai ir mobilios elektroninės laikmenos. Šiomis dienomis SD kortelių talpa svyruoja nuo 8 MB iki 32 GB. Didesnės nei 4 GB SD kortelės yra gaminamos naudojant SDHC technologiją. Populiariausia duomenų perdavimo magistralė atminties kortelėse naudojant mikrovaldiklius - SPI. VHDL programinė kalba leidžia aprašyti kaip ir skaitmeninės grandinės darbą, taip ir struktūros tipą, ji naudojama daugelyje sistemų, modeliuojant skaitmenines grandines, projektuojant programuojamas logines integralines mikroschemas ir bazinius matricos tipo kristalus. Vienas svarbiausių aparatūros aprašymo kalbų privalumų – galimybė sintezuoti grandinę arba sistemą į programuojamą įrenginį (FPGA). Mikrovaldikliai naudojami įrenginiuose turinčiuose automatinį valdymą. Sistemos kurios, naudoją atskirą mikroprocesorių, atmintį, įėjimo ir išėjimo įrenginius vis rečiau naudojamos, nes mikrovaldikliai atlieka tas pačias funkcijas už mažesnę kainą ir sutaupo schemose vietos.
- Ištyrus failinių sistemų realizacijų skirtų mikrovaldikliams savybes ir funkcijas paaiškėjo: tinkamiausia tyrimui yra Bill Greiman failinės sistemos realizacija, nes apie šia sistemą surinkta mažai duomenų. Kitos realizacijos buvo atmestos dėl nepopuliarių ar brangių mikrovaldiklių naudojimo, standžiųjų diskų naudojimo atsižvelgiant į jų kainą ir dydį.
- Atliekant failinės sistemos skirtos mikrovaldikliui tyrimą, testavimas buvo vykdomas dvejose aplinkose. *Arduino* platformoje pasiekti didžiausi duomenų perdavimo greičiai - skaitymo 276,86 KB/s ir rašymo 331,90 KB/s (naudojant 16 MHz mikrovaldiklio taktinį dažnį). Naudojant FPGA programuojamą matricą su integruotu programiniu AVR mikrovaldikliu pasiektas maksimalus stabilus taktinis dažnis 64 MHz, tai leido duomenis rašyti 264,02 KB/s ir skaityti 415,54 KB/s greičiu (naudojant programiškai realizuotą SPI sąsają).

REKOMENDACIJOS

Apibendrinus tyrimo rezultatus galima pateikti tokias rekomendacijas:

Tyrėjams

1. Ateityje atliekant panašaus pobūdžio tyrimą, realizuoti programiniame mikrovaldiklyje aparatūrinę SPI sąsają, kas leistų duomenų perdavimo greitį padidinti dar labiau.
2. Ateityje atliekant panašaus pobūdžio tyrimą, būtų vertingą taikyti kitą aparatūros aprašymo kalbą, taip sistemai suteikiant didesnes funkcines galimybes.

Kauno Technologijos Universitetui Informatikos fakultetui

1. Ateities tyrėjams būtų palanku naudoti kitų gamintojų FPGA programuojamąsias matricas, taip leidžiant tyrimui įgyti įvairialypiškumą.

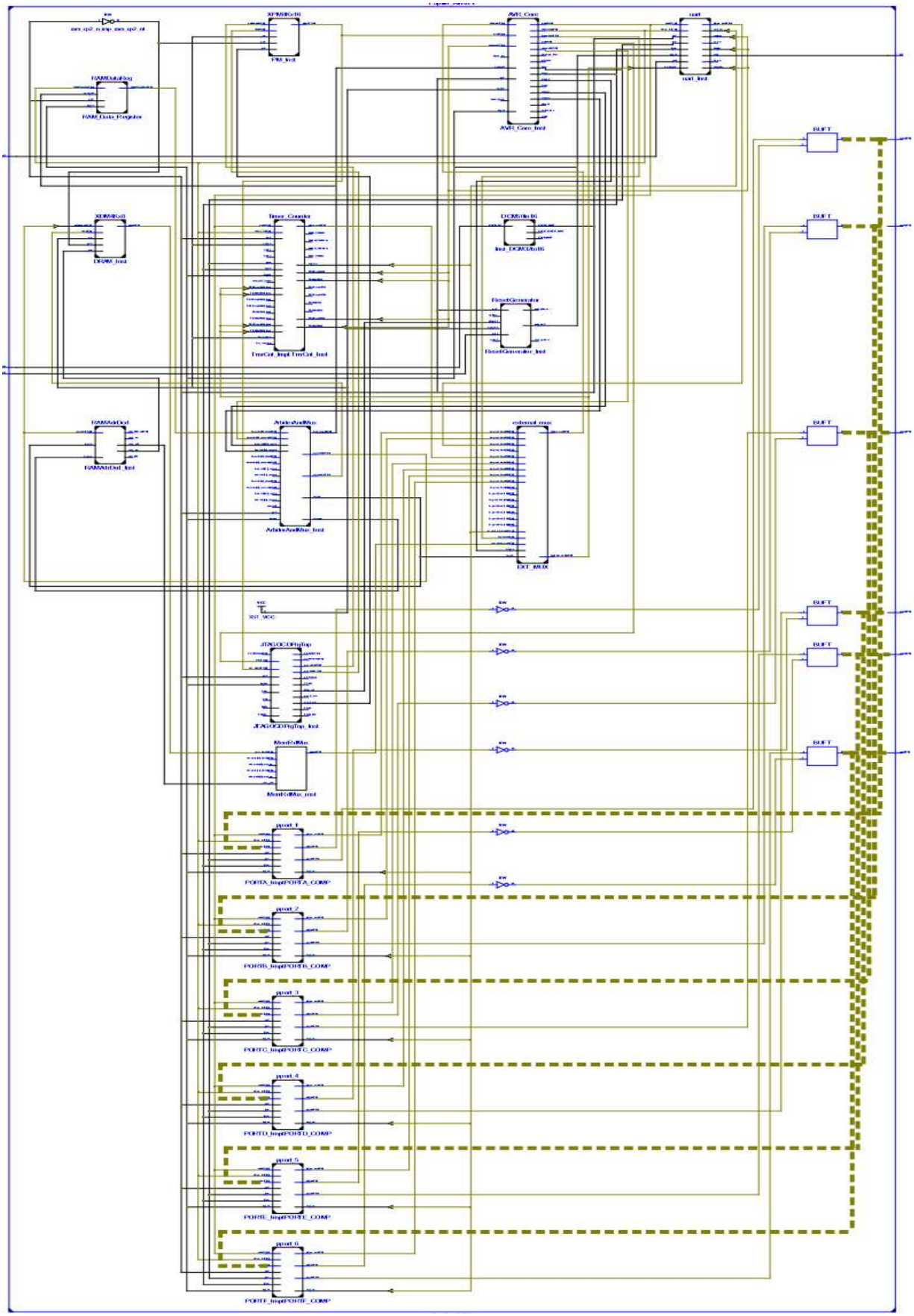
LITERATŪRA

1. Abraitis V., Bareiša E., Benisevičiūtė R., (2003) Programuojamų lustų testavimo metodai. Elektronika ir elektrotechnika.
2. Analog Devices Failinės sistemos realizacija mikrovaldikliui [žiūrėta 2011-03-15]. Prieiga per internetą: http://www.analog.com/static/importedfiles/application_notes/EE289.ADSP.BF533.Rev1.Feb.2006.pdf
3. Arduino oficialus tinklalapis [žiūrėta 2011-03-12]. Prieiga per internetą: http://www.arduino.cc/
4. Arnold K., Embedded Controller Hardware Design (2001). ISBN: 1-878707-52-3
5. Brian Jenkinson, Sammes, A. J. (2000). Forensic Computing: A Practitioner's Guide (Practitioner Series). ISBN 1-85233-299-9
6. FatFs, Failinės sistemos realizacija mikrovaldikliui [žiūrėta 2011-03-20]. Prieiga per internetą:http://elm-chan.org/fsw/ff/00index_e.html
7. Greiman B., Failinės sistemos realizacija mikrovaldikliui [žiūrėta 2011-02-16]. Prieiga per internetą:http://code.google.com/p/sdfatlib/
8. Grigaitis D., (2009) Programuojamų loginių matricų pradžiamokslis
9. Jusas V., Bareiša E., Šeinauskas R., (1997). Skaitmeninių sistemų projektavimas VHDL kalba. ISBN 9986-13-580-X
10. Microsoft Extensible Firmware Initiative FAT32 File System Specification, rev. 1.03, Dec 6, 2000.
11. Microsoft Help and Support. Maximum Partition Size Using FAT16 File System - Revision: 3.3. 2007 sausis [žiūrėta 2011-02-09]. Prieiga per internetą: http://support.microsoft.com/kb/118335.
12. Microsoft Resources for IT Professionals [žiūrėta 2011-04-25]. Prieiga per internetą: http://technet.microsoft.com/en-us/library/cc750198.aspx
13. Pedroni V. A., (2004). Circuit Design with VHDL. ISBN 0-262-16224-5
14. Pong Chu P., (2006). RTL hardware design using VHDL. ISBN-13: 978-0-471-72092-8
15. Pong Chu P., (2008) FPGA Prototyping by VHDL Examples: Xilinx Spartan-3 Version
16. Riegel R., Failinės sistemos realizacija mikrovaldikliui [žiūrėta 2011-03-20]. Prieiga per internetą: http://www.roland-riegel.de/sd-reader/
17. SanDisk Secure Digital Card Product Manual Version 1.9 SanDisk Corporation
18. SanDisk. Secure Digital Card Product Manual - Revision 1.7, September 2003
19. SD Specifications Physical Layer Simplified Specification Version 2.0, September 25, 2006

20. SDFileSystemVersion Failinės sistemos realizacija mikrovaldikliui [žiūrėta 2011-03-20]. Prieiga per internetą: http://www.sfcompiler.co.uk/wiki/pmwiki.php?n=SwordfishUser.SDFileSystemVersion
21. Thomas E. Donald, Moorby R. Philip, The Verilog hardware descriptiion language, fifth Editon. (2002). ISBN: 0-306-47666-5

PRIEDAI

1 priedas.



2 priedas.

Failinės sistemos skirtos mikrovaldikliui sintezės gauti rezultatai:

Gradinės komponentų ataskaita:

```
Advanced HDL Synthesis Report

Macro Statistics
# FSMs                                          : 1
# Adders/Subtractors                            : 13
 16-bit adder                                   : 4
 16-bit subtractor                              : 3
 4-bit adder                                    : 1
 5-bit adder                                    : 1
 7-bit adder                                    : 1
 8-bit adder                                    : 1
 8-bit addsub                                   : 2
# Counters                                      : 9
 10-bit up counter                              : 2
 12-bit up counter                              : 1
 16-bit up counter                              : 1
 2-bit up counter                               : 4
 4-bit up counter                               : 1
# Registers                                     : 1131
 Flip-Flops                                     : 1131
# Comparators                                   : 5
 8-bit comparator equal                         : 3
 8-bit comparator not equal                     : 2
# Multiplexers                                  : 1
 16-bit 8-to-1 multiplexer                      : 1
# Priority Encoders                             : 1
 5-bit 1-of-10 priority encoder                 : 1
# Xors                                          : 41
 1-bit xor2                                     : 31
 1-bit xor3                                     : 9
 8-bit xor2                                     : 1
```

Energijos ir temperatūrų ataskaita (naudojant *Xilinx XPower Analyzer*):

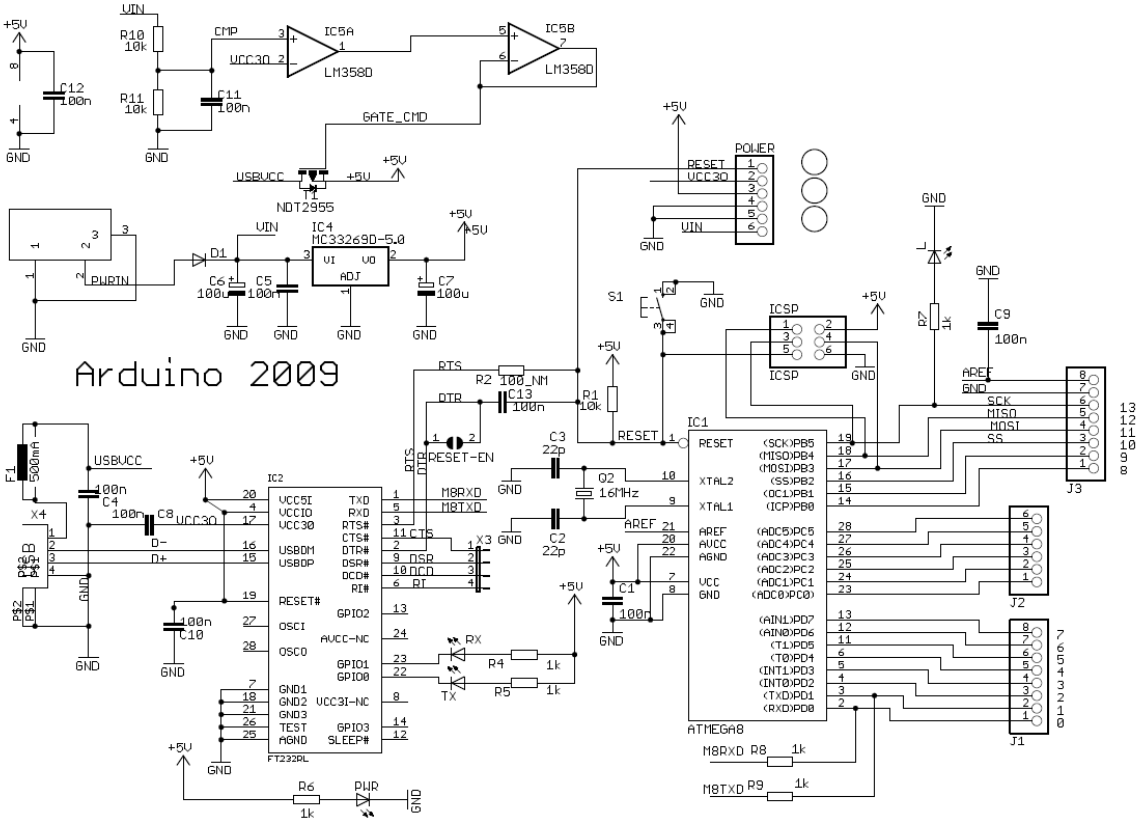
On-Chip Power Summary				
On-Chip	Power (mW)	Used	Available	Utilization (%)
Clocks	24.10	1	---	---
Logic	2.02	2415	11776	21
Signals	8.93	3160	---	---
IOs	0.02	52	372	14
BRAMs	15.52	10	20	50
DCMs	24.72	1	8	13
Quiescent	36.92			
Total	112.24			

Thermal Summary	
Effective TJA (C/W)	19.4
Max Ambient (C)	82.8
Junction Temp (C)	27.2

Power Supply Summary			
	Total	Dynamic	Quiescent
Supply Power (mW)	112.24	56.58	55.66

3 priedas.

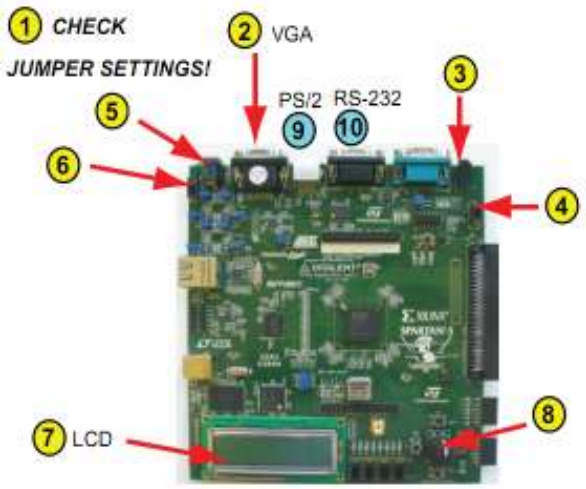
Arduino platforma (schema ir nuotrauka paimta iš oficialiaus arduino tinklalapio www.arduino.cc)



4 priedas.

FPGA programuojamoji matrica

Šiame tiriamajame darbe naudojama *Xilinx Spartan-3AN Starter Kit* programuojamoji matrica, todėl būtina aptarti šios plokštės teikiamas funkcijas ir esmines savybes. FPGA išvadai suteikia projektuotojui galimybę prijungti išorinius įrenginius, taip atliekant kelis procesus vienu metu.



1. Prieš pradedant darbą su FPGA lustu, būtina kas kartą patikrinti trumpiklių (angl. *jumpers*) nuostatą, jie reikalingi tinkamai lusto darbui.
2. VGA jungtis, skirta prijungti vaizduokliui arba projektoriui.
3. AUDIO jungtis, skirta prijungti ausinėms arba kolonėlėms.
4. SUSPEN ir RUN jungiklis.
5. Lusto maitinimo lizdas.
6. Įjungimo jungiklis.
7. LCD ekranas, rodantis įvairias informaciją ir instrukcijas.
8. Potenciometras

5 priedas.

Baigiamajame magistro darbe naudota programinė įranga

- Windows ir Linux OS
- ModelSim PE Student Edition 10.0
- Xilinx ISE Design Suite 13.1 webpac license
- Notepad++
- OpenOffice ir Microsoft Office
- DIA
- Arduino IDE
- AVR-GCC konverteriai
- AVR-Studio
- WinAVR
- Putty
- RealTerm v 2.0