KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
PROGRAMŲ INŽINERIJOS KATEDRA

Vienlusčių tinklo architektūrų tyrimas

Magistro darbas

Darbo vadovas

dr. R. Damaševičius

Kaunas, 2007

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
PROGRAMŲ INŽINERIJOS KATEDRA

Modestas Čepaitis

Vienlusčių tinklo architektūrų tyrimas

Magistro darbas

Recenzentas
prof. V.Jusas
2007 - 05 -

Vadovas
dr. R. Damaševičius
2007 - 05 -

Atliko
IFM-1/5 gr. stud.
Modestas Čepaitis
2007 – 05- 26

Kaunas, 2007

SANTRAUKA

“International Technology Roadmap for Semiconductors” (ITRS) teigia, kad baigiantis dešimtmečiui bus pagamintas 50 – 100 nm lustas susidedantis iš maždaug 4 bilijonų tranzistorių operuojančių 10Ghz. Taigi tokių parametrų sistemas pagaminti yra nelengva, tranzistorių skaičiaus didėjimas taip pat didina lusto dydį, lusto sudėtingumą bei sunkina lusto integruojamumą. Be to vienlustės sistemos naudoja bendras magistrales bendrauti su kitais lustiniais resursais. Šios magistralės yra nedalomos, todėl ateityje toks komunikavimo metodas taps vis didesne problema gaminant lustus susidedančius iš bilijonų tranzistorių. Šiems tikslams spręsti ir buvo pasiūlyta vienlusčių tinklo paradigma. Vienlusčių tinklų paradigma siūlo galimus komunikavimo infrastruktūrų sprendimus, susiduriant su vis sudėtingesnėmis sistemomis, bei padeda trumpinanti lusto pagaminimo laiko periodą. Šiame darbe naudojama atkartojimo technologijos, kuriant bendrini komponentą vienlusčių tinklams. Metaspecifikacija, sukurta naudojant preprocesorinį atkartojimo metodą

Research in Network on Chip architectures

SUMMARY

According to the International Technology Roadmap for Semiconductors (ITRS), before the end of this decade we will be entering the era of a billion transistors on a single chip. It is being stated that soon we will have a chip of 50- 100 nm comprising around 4 billion transistors operating at a frequency of 10 Ghz. However, it has been observed that as the system grows, so does the complexity of integrating various components on a chip. The major threat toward the achievement of a billion transistor chip is poor scalability of current interconnect structure of today's SoC. In order to cope with growing interconnect infrastructure, the "Network on chip (NoC)" concept was introduced. NoCs present a possible communication infrastructure solution to deal with increased design complexity and shrinking time-to-market. It is clear that NoCs can potentially become the preferred interconnection approach for SoCs being developed in a near future. This paper discusses the impact of the reuse of NoC components methods, for parameterize and test these systems. There is preprocessing type reusing being used, to create a router metaspecification of router for the 2d torus interconnect network.

TURINYS

PAVEIKSLŲ SĄRAŠAS

LENTELIŲ SĄRAŠAS

ĮVADAS

Ateities vienlustėse sistemose bus integruota šimtai resursų, kurie bus pagaminti iš bilijonų tranzistorių. Tokios sistemos reikalauja komunikavimo standartų, leidžiančių pasiekti dešimčių Gbits/s duomenų siuntimo greitaveiką, taip pat tenkinančių sistemos dalumo, bei atkartojimumo mechanizmų, ir žinoma neužtrunkančių per ilgai, kad išleisti tokią sistemą i prekybą. Kaip teigiama [1], [10] literatūros šaltiniuose problemų sprendimui siūloma integruojamieji vienlusčių komunikavimo mechanizmai - vadinami vienlusčių tinklais.

Vienlusčių tinklas tai struktūriškai išdėstytas maršrutizatorių rinkinys, kur maršrutizatoriai yra sujungti tarpusavyje, bei sujungti su duomenų vienlusčių sistemų tipo resursais. Literatūros šaltiniuose [11], [17], [6] apibūdinta nemažai tokių sistemų variantų, kur pagrindinis ir paprasčiausias yra 2D MESH. Kiekvienas resursas prijungtas prie atskiro maršrutizatoriaus, o maršrutizatoriai sujungti tarpusavyje su artimiausiais šalia esančiais maršrutizatoriais. Resursai gali būti atminties blokai, ar skaičiavimų procesoriai, tai yra vienlustės sistemos kurioms reikia komunikuoti tarpusavyje.

Daugiaprocesoriniai vienlusčių tinklai tampa vis labiau populiariu sprendimu, nuolat greitesnių darbo parametrų reikalaujančiose sistemose. Vienas pagrindinių tokių sistemų aspektų yra komunikavimo posistemė. [5] Straipsnyje pateiktas tyrimo rezultatai parodo kad vienlusčių tinklai pasižymi geresne komunikavimo charakteristika, vykstant nuolatiniam duomenų perdavimui, palyginus su vienluste sistema, kurioje komunikavimas vyksta centrine magistraline jungtimi, bandymo metu, kai tarpusavyje buvo sujungti aštuoni resursai. Esant mažesnėms apkrovoms, centrines magistralinės jungties darbo parametrai bus geresni, nei vienlusčių tinklo. Tokie magistralinės jungties privalumai galioja tokiose sistemose, kuriose yra daugiau nei 16 resursų .[5]

Komunikavimo posisteme turėtų tenkinti tokius reikalavimus kaip mažos galios vartojimas, dalumas - didinant ar mažinant posistemės dydį, bei lengvai modifikuojamumas norint pakeisti atskira sistemos dalį. Norint pasiekti kuo geresnius tokių parametrų rodiklius, ir pagaminti vienlusčių tinklą, pasitelkiama programų gaminių linijų kūrimo technologijos.

Gaminių linijų kūrimo architektūros pagrindinis principas yra gaminio variantiškumas. Pagrindinis gaminių linijų efektyvumas pasirodo norint patestuoti sistemas. Tai labai efektyvi strategija plaukianti iš programinio kodo atkartojimo mechanizmų panaudojimo.

Pritaikius atkarojimo technologijų galimybes galima potencialiai pagerinti programų produktyvumą, sutrumpinti kūrimo procesą bei sumažinti projekto kainą. Pasak literatūros šaltinio [2], dabartiniu metu konkrečios metodologijos, kuri sujungtų atkartojimą, kaip standartini kūrimo etapą, ir faktorių, kurie veikia atkartojimą, reikšmingumo lygmuo yra kontraversiškas. Kita vertus kodo atkartojimas galima vadinti objektiškai naudojamas programavimo priemones.

Šio darbo tikslas panaudojant metaprogramavimo, atkartojimo metodus, sukurti metaspecifikacją leidžiančia ištirti vienlusčių tinklų architektukūrų savybes. Kadangi maršrutizatorius turi didžiausią įtaką duomenų perdavimo parametrams vienlusčių tinkle, tyrimo tikslas sukurti tokio vienlusčių tinklo maršrutizatoriaus bendrini komponentą, ir keletą jo egzempliorių, kad ištirti tinklo parametrus kiekvieno egzemplioriaus panaudojimo atveju. Tam tikslui pasirinktas preprocesorinis atkartojimo metodas.

TECHNOLOGINIO LYGIO ANALIZĖ

1.1 Vienlusčių tinklų architektūros

1.1.1. Vienlusčių tinklų apžvalga

Elektronikos industrija nuėjo ilga kelia nuo to laiko kai pirmosios integrinės schemos buvo pagamintos 1950- ais. Galimas mikroschemų vystymosi planas toliau seka Moor'o taisykle su nenumanoma pabaiga.

Integrinių schemų evoliucijoje gaminiai tapo vis sudėtingesni. Tai labai padidino besirandantį plyšį tarp produktų galimybių ir technologijų jas įgyvendinant bei ploto, kuriame gali būti įgyvendinamos šios technologijos. Ši problema dar vadinama kaip „design-gap“.

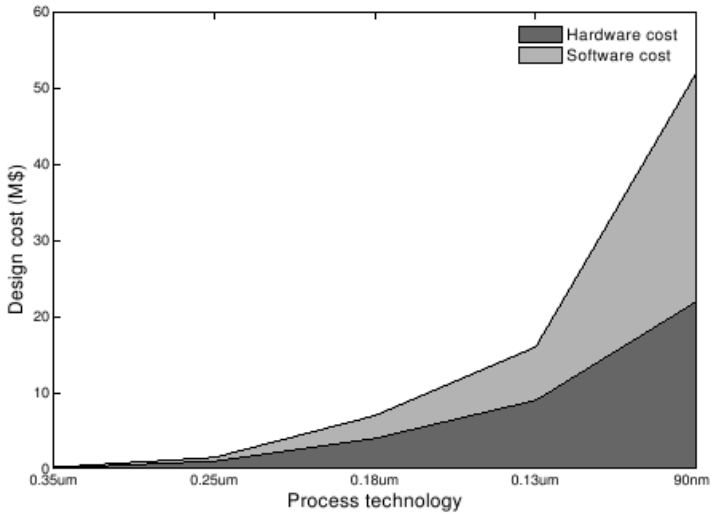
Naujos technologinės platformos ir kūrimo metodologijos turi padėti užpildyti šį plyšį. Techninės platformos tebėra labai panašios į tas kurios buvo pradėtos gaminti nuo senų puslaidininkių kompiuterių laikų ankstyvaisiais 1970 -aisias. Sistema tipiškai susideda iš vieno procesoriaus, galbūt iš dar kokios tai spartinančiosios dalies, pora tipų lokalios darbinės atminties, ir keleto papildomų įrenginių. Visi komponentai sujungti į viena magistralę. Tokio tipo platformos gali eiti tik tiek toli - kiek gali eiti tokio tipo industrija kuri jau yra ant naujos elektronikos dizainų eros slenksčio. Senosios platformos turi būti pakeistos kažkuo nauju, labiau dalomu.

Antros kartos platformos kurios gaminamos šiandien yra tik perdarytos pirmosios kartos produktai. Skirtumas tik tas, kad procesorius dažniausiai yra pakeičiamas mikro valdiklių ir skaitmeninių signalų procesoriumi. Negalima apibūdinti tokio tipo platformos daugiaprocesorine kadangi tebėra vienos magistralės ribojimas.

Dabartinė kryptis yra pagaminti heterogenišką, įvairiarūšę daugiaprocesorinę sistemą. Tokia sistema susidėtų iš kelių procesorių, o tai atneša labai daug naujų sunkumų kol tokias sistemas pagaminti. Tokie sunkumai kaip: sujungimas, komunikavimas ir programinė įranga yra tik kelios sritys kurias dar reikia gerai ištirti, pradedant gaminti tokias platformas.

Tuo metu žiūrint į vienlusčių sistemų sprendimus sudėtinėms sistemoms, tokioms kaip interneto tinklų sąsaja , mobiliųjų telekomunikacijų bazinėms stotims, sudėtingos problemos sprendžiamos su labai daug veikiančių mazgų, kurie jungia tinklą. Daugiaprocesoriniai mazgai neišvengiamai paveiks paketų perdavimą nuo to laiko kai dalinamoji atmintis neatskiriamai blogai tada kai tai prieina dalinimosi laikas.

Vienlustėms sistemoms yra reikšmingas programinės įrangos kūrimo kainos klausimas. Programinės įrangos kūrimo kainą auga greičiau nei pačių sistemų gaminimas.



1 pav. Sistemų geležies ir programinės įrangos kainų kitimo grafikas [7]

1.1.2. Komunikavimas tinklu

1.1.2.1 Tinklo (OSI)

OSI –„The Open Systems Interconnection Reference“ modelis susideda iš septynių pagrindinių lygmenų, kurie apibūdina tinklo veikimą. Modelis neapibūdina tiksliai kaip tūri būti sukurta sistema, bet atlieka koncepcijos vaidmenį. Kiekvienas lygmuo susideda iš atskirų ir tik tam lygmeniui būdingų charakteristikų. Junginys šių sluoksnių yra žinomas kaip protokolų registras ir gali būti sukurtas, įgyvendintas programiškai ar techniškai.

Žemiau pavaizduotoje lentelėje, trumpas kai kurių tinklo lygmenų sąrašas . vienlusčių tinklo sistema dažniausiai kuriama tokia, - kurią galima apibūdinti pirmais penkiais lygmenimis:

1 lentelė. Tinklo OSI lygiai [15]

Pirmas lygmuo	Fizinis lygmuo	Elektrinių ir fizinių savybių specifikavimas.
Antras lygmuo	Duomenų sąsajos lygmuo	Duomenų perdavimo ir klaidų nustatymo metodologijos specifikavimas.
Trečias lygmuo	Tinklo lygmuo	Tinklo maršrutizavimo, kokybės specifikavimas.
Ketvirtas lygmuo	Transporto lygmuo	Paketų siuntimo ir klaidų taisymo specifikavimas.
Penktas lygmuo	Sesijos lygmuo	Jungties sukūrimo ir nutraukimo specifikavimas/
Šeštas lygmuo	Paruošimo lygmuo	Duomenų suspaudimo ir patikimumo specifikavimas.
Septintas lygmuo	Sąsajos lygmuo	Interfeiso tarp tinklo ir panaudojimo specifikavimas.

Vienlusčių tinklo topologija apibūdina jungimosi metodą tarp tinkle esančių komponentų. Topologija gali būti atvaizduota grafu, kur viršūnes yra tinklo mazgai o grafo briaunos yra sąsajos tarp viršūnių. Mazgai yra elementai kurie gali priimti, pagaminti ar išsiųsti žinutes į vienlusčių tinklą.

1.1.2.2 **Rankų paspaudimo protokolas**

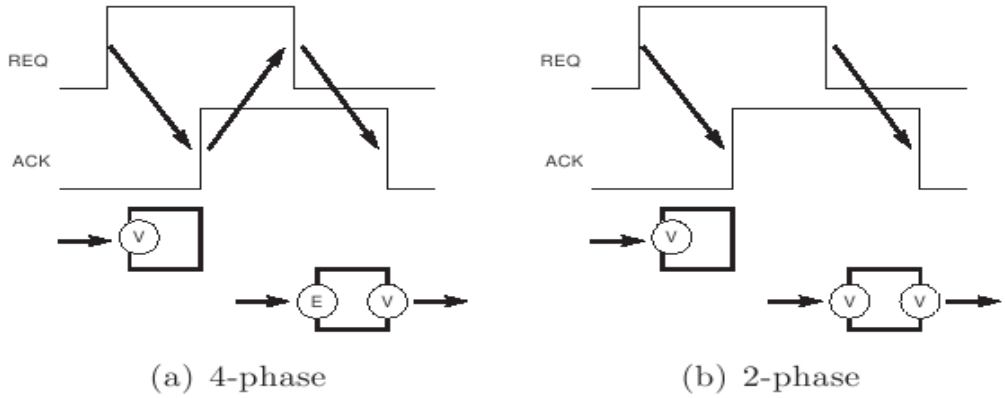
Pagrindinis 2 lygmens duomenų perdavimo elementas yra “Rankų paspaudimo“ protokolas. Rankų paspaudimo protokolas yra papildoma dalis komunikuojant dviems elementams (nėra būtinas visose užbaltose sistemose). Įprasta, kad komunikavimo iniciatorius yra vadinamas šeimininku, o komunikavimo adresatas vergu. Keturių žingsnių „rankų paspaudimo“ inicializavimas:

1. Patvirtinimo užklausimas.
2. Patvirtinimo gavimas.
3. Gavimo patvirtinimas.
4.Patvirtinimo užklausos patvirtinimas.

Dviejų žingsnių „rankų paspaudimo“ inicializavimas:

1. Užklausos gražinimas.
2. Grąžos patvirtinimas.

Šis 2 fazių “rankų paspaudimo“ protokolas skirtas padidinti perdavimo greitaveiką palyginus su 4 fazių “rankų paspaudimo” protokolu.



2 pav. Dviejų žingsnių „rankų paspaudimo“ inicializavimas.

1.1.2.3 **Maršrutizavimas**

Maršrutizavimo terminas remiasi duomenų perdavimu tarp dviejų tinklo resursų naudojantis papildoma įranga. Tai būtų trečiasis OSI lygis, aukščiau apibūdinančioje lentelėje 0.1 . Yra trys pagrindiniai maršrutizavimo tipai:

1. Nukreipiamasis maršrutizavimas.

Šis mašrutizavimo tipas dažnai apibūdinamas kaip „hot-potatoe“ maršrutizavimas. Kai kiekvienas mazgas stengiasi kuo greičiau atsikratyti duomenų paketu (kaip laikant rankose karštą bulvę). Tai nedaug “kainuoja“ resursų sąskaitai, kadangi nereikalingas duomenų buferis kiekvienam mazgui, bet toks maršrutizavimas nepalaiko duomenų „datagram“ paketų siuntimo. Jei duomenų paketas negali būti priimtas adresato, jis yra nukreipiamas grįžti atgal po kurio tai laiko. Toks veikimas turi kelis labai didelius ribojus ir nepakankamus. Duomenys gali ateiti nesurikiuoti, neeiliški, todėl rikiavimo mechanizmas turi būti naudojamas.

2. „Išsaugok ir perliūsk“ maršrutizavimas.

Maršrutizavimo tipas, kuris naudoja duomenų buferius kiekviename žingsnyje perduodant paketus. Šie buferiai (FIFO) gali būti sukurti iš įvairiausių algoritmų siekiant pagerinti jų veikimą.

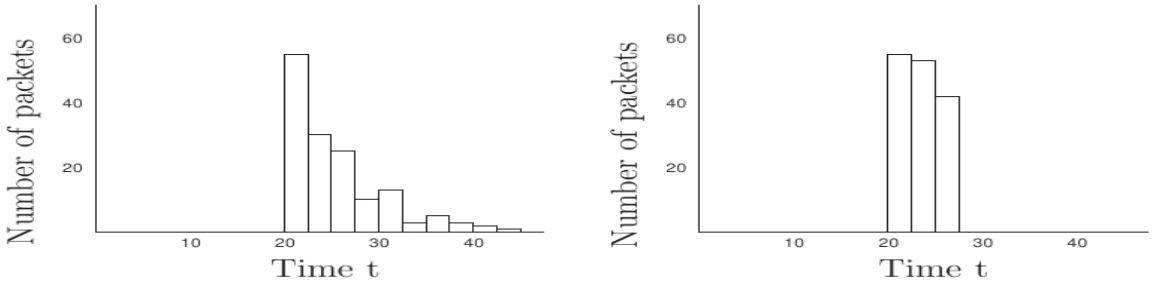
3. Virtualaus rato maršrutizavimas.

Virtualaus rato maršrutizavimas sukuria nuolatinį ryšį tarp funkcionuojančių vienetų. Maršrutas gali būti nustatomas naudojant centralizuotas maršrutizavimo direktyvas. Kad išvengti maršrutų susikirtimo ir klaidų, yra kuriami virtualūs maršrutai, skirtingos paketo dalys skiriasi savo dydžiu ir laiku.

OSI standarto trečiajame lygyje, yra ir daugiau maršrutizavimo pasirinkimų, tipų. Kad išskirti juos nuo kitų maršrutizavimo metodų jie yra vadinami aukšto lygio maršrutizavimo metodais. Yra labai daug galimybių tokių kaip multiplikacinis maršrutizavimas duomenų srauto kontroliavimas ir netgi. Šie maršrutizavimo metodai tinkle dažniausiai naudojami kai iškyla neeiliškas paketų perdavimas.

1.1.2.4 **Ryšio patikimumas**

Du pagrindiniai (OSI sluoksniai) yra skirti ryšio patikimumui apibūdinti. Jei minimalią perdavimo kokybę galima garantuoti bet kuriomis tinklo komunikavimo sąlygomis, - tai tokios sąlygos yra vadinamos garantuotu duomenų srautu GB. Jei tai yra neįmanoma ir sistema gali perduoti duomenis geriausiomis sąlygomis tai tos sąlygos yra vadinamos geriausiu duomenų srautu - BE



3 paveikslėlis. BE ir GB duomenų siuntimo vėlinimo elgsena. [15]

BE duomenų siuntimo sistemose vėlinimas tiesiogiai priklauso nuo tinklo apkrovimo. Iš grafiko matome kad ir GB duomenų siuntimo sistemoje vėlinimas yra tas pats bet su

palyginti labai mažu variavimu. GB sistema dirba tik tada kai yra kažkokia tai paketų siuntimų prioritetų sistema. Ši sistema turi paskirstyti paketų svarbumą, aukštą ar žemą. Paprasta prioritetų yra pavojinga, todėl kad žemo prioriteto paketai kali būti visiškai užblokuoti tinkle aukšto prioriteto paketų siuntimu. Kitas GB trūkumas yra nenaudingas duomenų srauto apkrovimas, jei funkcionuojantys resursai nenaudoja pilno GB srauto. Esami NOC modeliai stengiasi suderinti vienoje sistemoje BE ir GB.

1.1.3. **Vienlusčių tinklų principai**

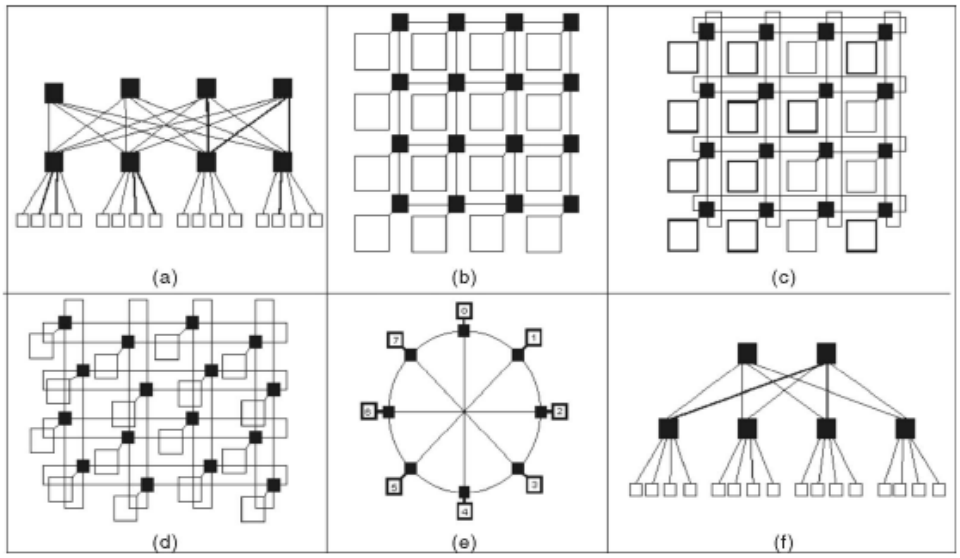
Daugiaprocesorinės vienlustės platformos iškyla kaip labai svarbi vienlusčių technologijos tendencija. Maitinimo ir laidų technologijos yra adaptuojamos į naujas vienlustes sistemas, jų mazgų įjungimą. Šitie centro komunikavimo susisiekimo produktai yra charakterizuojami pagal daug skirtingų parametrų kur vieni iš pagrindinių yra vėlinimas, duomenų perdavimo srautas, energijos sunaudojimas, ploto reikalavimai. [16]

Šios problemos sprendžiamos komponentinėse daugiaprocesorinėse SOC platformose pasitelkiant įvairaus susisiekimo topologijų tipus tokioms platformoms. Bet apjungimas kelių komponentų į vientisą sistema sukelia daug naujų problemų.

Viena pagrindinių vienlusčių technologijos problemų yra globalus nepadalinimas, nesumažinamas laidų vėlinimas. Globaliais laidais keliauja signalai mikroschemoje, bet šie laidai dažniausiai nesidalina ilgio atžvilgiu kartu su technologijos tobulėjimu skirtingai nei ventilių vėlinimai. Pavyzdžiui labai mažame “Submicron'e“ 80 % ar daugiau vėlavimų yra jungiamuosiuose laiduose. Taigi tokios problemos yra sprendžiamos pasitelkiant tinklinius sprendimus. Dažniausiai naudojamas vienlustėje sistemoje jungimas yra architektūra, kur naudojama bendra magistralė, kur visi komunikuojantys įrenginiai dalinasi bendru kanalu. Tokios technologijos privalumas yra paprasta topologija, maža plotas, ir erdvumas. Kad ir kaip bebūtų palyginus ilgai magistralės linijai, būdingas parazitinis laidumas ir talpumas gali būti gan aukšti. Be to prie magistralės pridėjus vis naujų IP blokų padidėja parazitinis talpumas tuo metu didindamas vėlinimą. Vienas tokios problemos sprendimo metodų yra padalinti tą magistralę į keletą segmentų ir pritaikyti hierarchinę architektūrą, bet ir šis sprendimas turi daug trūkumų tokiose sistemose. SOC gali susidėti iš šimtų IP blokų, ir

magistralinėje sistemoje tai gali pasireikšti kaip butelio kaklelio (Bottleneck) problemos kadangi visi įrenginiai naudojasi viena magistrale. Apeiti šias problemas buvo pradėta integruoti IP blokus į tinklo sistemas. Naujas modelis leidžia bendrauti procesoriniams elementams.

Tokiose tinklo sistemose komunikavimas tarp IP blokų gali vykti paketų forma. Susiduriama kad NOC yra panašus į aukšto lygio jungiamąją architektūrą. Bendras panašumas tokios rūšies architektūrose yra IP blokų komunikavimas „protingų“ komutatorių pagalba. [16]



4 pav. Vienlusčių tinklų topologijos. (a) SPIN, (b) CLICHE, (c) Torus, (d) Folded torus, (e) Octagon, (f) BFT.[16]

NOC platformos kūrimas yra sudėtingas procesas dar iki šiol tyrimų sritis yra labai plati. Yra didelis skaičius sukurtų NOC tipų ir sprendimų. Daugelis šitų tipų yra paremtos sistemų veikimu kurį labai sunku nustatyti kūrimo metu.

Taigi šio tipo platformas pirmiausia stengiamasi daryti tokias kad būtų galima jas veiksmingai panaudoti ant silkoninio paviršiaus, minkštais, mažais komponentais su aukštais veikimo parametrais. [7]

Tinklas yra rinkinys sujungtų tarpusavyje mazgų kuris leidžia nesudėtingą duomenų dalinamasi tarpusavyje. Tinkluose su komutatoriais daugialypiai mazgai yra sujungti tarpusavyje ir jie gali bendrauti vienas su kitu. Tuo būdu priėjimas prie siuntimo kanalo kokiu tai nors būdu dinamiškai paskirstomas visiems mazgams. Dažniausiai naudojami bendri metodai tai įvykdyti yra ratu maršrutizavimas ratu ir paketų komunikavimo metodai.

Išskyrus tinklo greitaveika, užklausos laiką yra dar daug kitų tinklo ypatybių charakterizuoti tinklo jungtis: jungtis gali būti patikimas ar nepatikimas, tai yra kai tinklu siunčiami duomenys gaunami adresato be jokių praradimų. Be to informacijos siuntimas gali būti nesujungiamas ar sujungiamas. Pirmu atveju informacija pasiunčiama gavėjui tik tada kai duomenis praeina jungimosi funkcijas. [11]

NOC veikimo charakteristikos

- Apkrova – apibūdina rekomenduojamą duomenų kiekį kuris gali būti persiunčiamas per laiko vienetą. Tai dažniausiai yra skaičiai nuo 0 (dera duomenų) iki 1 (duomenų dydis).

- Duomenų perdavimo greitis – nusako duomenų kiekį kuris buvo persiustas per laiko vienetą. Šis parametras gali priklausyti nuo duomenų šaltinio ir gavėjo padėties.

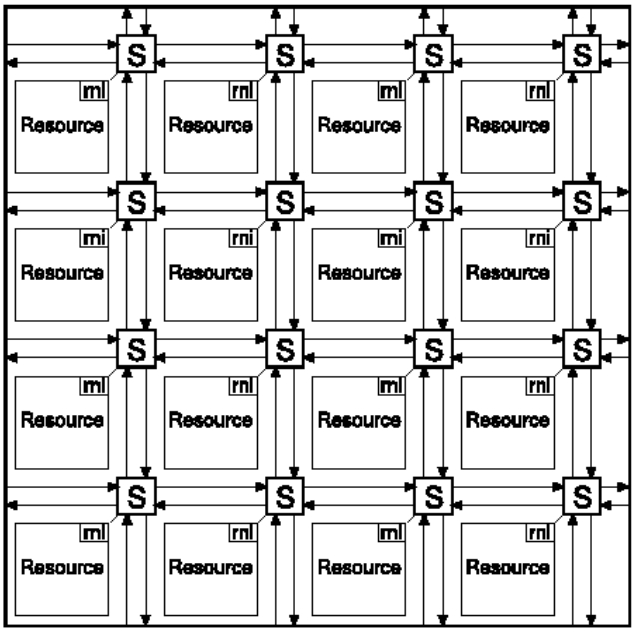
- Vėlinimas– nusako laiko trukmę per kurį paketas nueina nuo siuntėjo iki gavėjo.

- Paketų praradimas – yra skirtumas tarp paketo dydžių kurį mazgas norį išsiųsti ir tikro paketo dydžio kurį pavyksta išsiųsti.

- Sistemos dalumas – apibūdina kaip gerai patikimai jungtys dirba didinant mazgų kiekį prijungtų prie tos jungties. [5]

1.1.4. **2D MESH**

Tai paprastos struktūros, įprasto dviejų dimensijų topologijos architektūra. Reguliariame MESH išplanavime mikroschemos padalinamos į lygias tokio pačio dydžio blokus. Kiekviename bloke viena IP šerdis ir koks nors vartotojo pasirinktas loginis elementas. Šitie blokai sujungti laidais. Nutiesti tokius laidus mikroschemoje yra nesudėtinga kaip ir įprastoje SOC struktūroje.



5 pav. MESH NOC [11]

Mesh pagrindinės, būtinos tinklo charakteristikos:

- Resursų ir komutatorių kiekis vienodas
- Komutatorius sujungtas su keturiais komutatoriais ir su vienu resursu
- Resursas sujungtas su vienu komutatoriumi

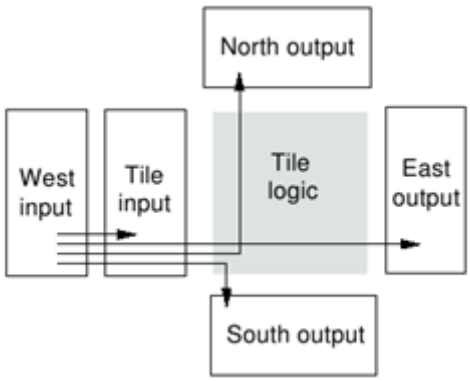
Kiekvienas blokas turi tinklo resurso bloką, , kuris jungia bloko loginę, loginio elemento, dalį su komutatoriumi. Komutatorius atlieka visas kitas informacijos komutavimo - siuntimo ar gavimo operacijas. Reikia pažymėti, kad įprasta struktūra leidžia pasirinkti

tiesioginį adresavimą, kuris veda prie paprastesnių maršrutizavimo algoritmų ir paprastesnės komutatorių sudėties.

Dar vienas naudingas rezultatas gautas iš unikalaus magistralių tiesimo ir trumpų laidų bei dviejų dimensijų dizaino: Elektriniai parametrai kaip priešingybė yra prognozuojami ir kontroliuojami. Be to sujungimai gali būti prievarta optimizuojamas, kad būtų naudojama mažiau galios ar kokie specifiniai vėlavimai reikalingi.

1.1.4.1 **Komutatorius**

Komutatorius susideda iš penkių įėjimo valdiklių ir penkių išėjimo valdiklių išdėstytų aplink bloko loginį elementą – vienas įėjimas ir vienas išėjimas į kiekvieną kryptį (šiaurės, rytų, pietų vakarų) ir po vieną bloko I/O. Paveikslėlyje 3.6 pavaizduota tik vakarų valdiklis ir jo jungtys su bloko įėjimu ir kitų valdiklių išėjimais.



6 pav. MESH komutatorius [11]

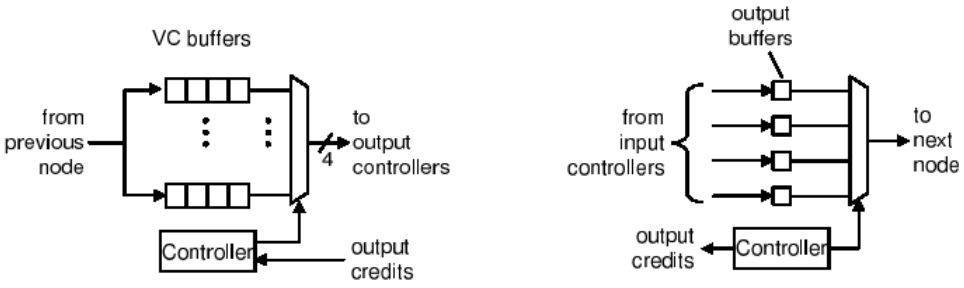
Kontrolės laukas turi pasiruošimo signalą kuris yra siunčiamas gavėjui jei tinklas yra pasiruošęs priimti duomenų informacijos vienetą.

Srauto kontrolė priklausomai kontroliuojama komutatorių. Dažniausiai, srautas stengiasi išvengti eilių, susigrūdimų, ar paketų praradimo tinkle

Mažiausias informacijos vienetas kuris gali būti persiustas šia struktūra yra srauto kontrolės skaitmuo flit paketas. Srauto kontrolė operuoja su šiais paketais kad užtikrinti aukštą perdavimo lygį, nedidelį vėlinimą ir naudinga laidų panaudojimą. Paketas susideda iš

kai kurios kontrolės informacijos ir duomenų apkrovos. Kontrolinės informacija yra paketo kelio duomenys ir virtualaus kanalo numeris. Tas virtualus kanalo numeris identifikuoja kanalą (loginį kelią iš siuntėjo į gavėją gautą tam tikrai jungčiai) leidžiantį emuliuoti sukamąjį komutavimo rėžimą paketų komutavimo tinkle. [11]

1.1.4.2**Maršrutizavimas.**



7 pav. Įėjimo kontroleris (kairėje) ir išėjimo kontroleris [11]

Kaip pavaizduota 8 pav. įėjimo valdiklis turi skirtingus įėjimo buferius kiekvienam virtualiam kanalui. Įeinantys “flit'ai“ yra saugomi atitinkamo kanalo buferio eilėje.

Maršrutizavimo informacijoje, kuri yra saugoma pradiniame perdavimo sesijos pakete, apibūdina vieną iš keturių ėjimo nuorodų (blokas, šiaure, ...). Loginio elemento kontrolė pasirenka paketus iš virtualaus kanalo eilės pasiuntimui toliau į išėjimo valdiklius. O tada vėl paketai yra statomi į eilę išėjimo valdiklyje.. Tai ir vaizduoja 3.7 pav. išėjimo valdiklis. Valdiklio logika sprendžia kuris įėjimo valdiklis išsiūs išėjimo nuorodą į sekantį mazgą.

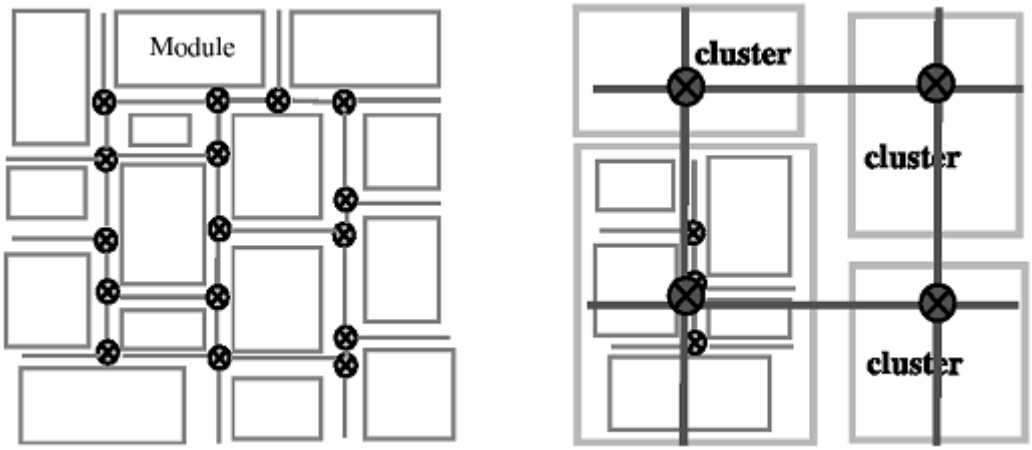
Tam kad vyktų grįžtamasis komutavimas paketų-komutuojamame tinkle, paketas turi virtualaus kanalo adresą savo turinyje. Jei virtualus kanalas yra naudojamas tik tam tikros siuntėjo-gavėjo poros, galima galvoti apie loginį kelią tarp jų. Geresnės darbo kokybės galima tikėtis garantavus resursų kaupimą tam kanalui išsaugotam.

Skirtingos duomenų srauto klasės SOC tinke gali būti adresuojamos pagal panašumus, bruožus kai yra perduodama daugiau pastovių duomenų (hight bandwidth video stream to

DSP) arba dinaminių, neprognozuojamo duomenų srauto. Pastovus srautas gali būti priskirtas virtualiam kanalui garantuoti reikiamą duomenų perdavimo greitį bet kuriuo atveju. Dinaminiai duomenys naudojami kai atsilaisvina komutatoriai.

1.1.4.3 **Kiti privalumai**

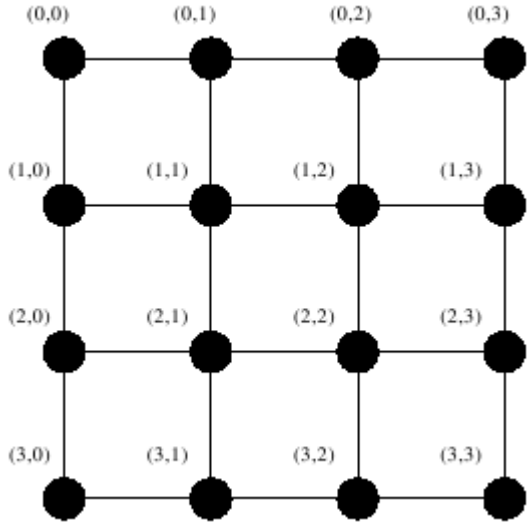
MESH topologija efektyviai naudingumas pasitvirtina sujungus susijusias dalis, kurios gali komutuoti priklausomai, palyginus greitai su kaimyninėmis tinklo mikroschemomis. Todėl gaunamas trumpas susisiekimo atstumas tik su keliais komutatoriais tarpusavyje. Du pavyzdžiai pavaizduoti 3.8 paveikslėlyje, reguliarus ir hierarchinis MESH.



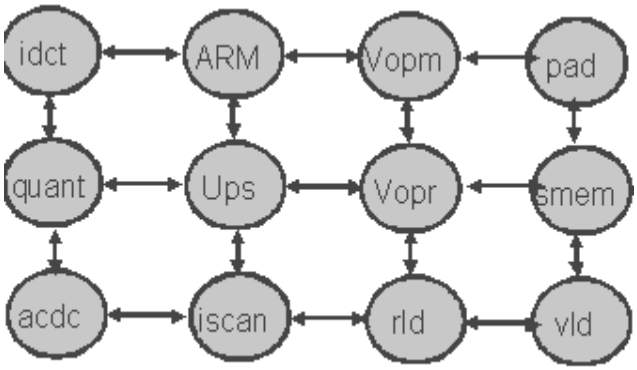
8 pav. Standartinis MESH (kairėje) ir hierarchinis MESH [11]

Standartinis MESH yra lengviau adaptuojamas SOC struktūros tikslams jei tik nėra tvirtų apribojimų IP blokų dydžiams. Kaip pavaizduota paveikslėlyje blokų dydžiais yra varijuojama. Komutatorių dėžutės yra įtaisytos kur bent dvi komutavimo linijos kertasi.

Hierarchinis MESH stengiasi komutuoti resursų blokus grupuodamas juos į vidinius vienetus sujungiant juos standartiniu MESH ir tuos gautus blokus jungiant globaliu MESH tinklu. Čia komunikavimas vyksta vienetiniuose blokuose.



9 pav. MESH komutatorių galimas adresacijos variantas [17]

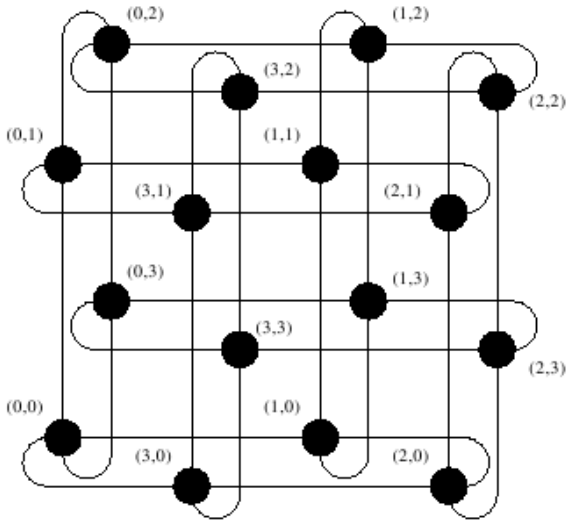


10 pav. Realiai realizuoto MESH tinklo schema:

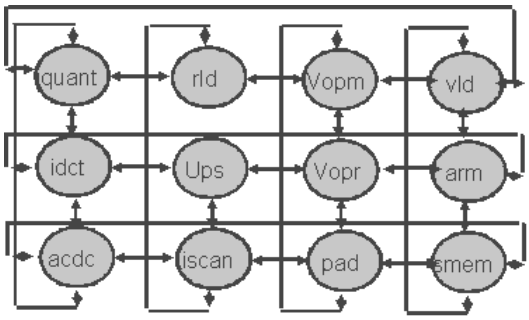
1.1.5. **2D – TORUS**

Torus topologija yra patraukli todėl kad gali būti projektuojama ir gaminama ant dviejų dimensijų (2D) ploto, taip kaip ir mech topologija. 2D torus susideda iš N = k^2 (k kvadratu) susikirtimo mazgų (komutatoriaus ir vartotojo naudojamo loginio elemento) kur k yra mazgų skaičius kiekvienoje dimensijoj.

Kiekvienas mazgas yra adresuojamas dviem skaitmenimis. Ir tas kiekvienas mazgas sujungtas su pora kanalų dvejomis kryptimis kurių mazgų adresai skiriasi tik +-1 mod k tiksliai vienu adreso skaitmeniu. [17]

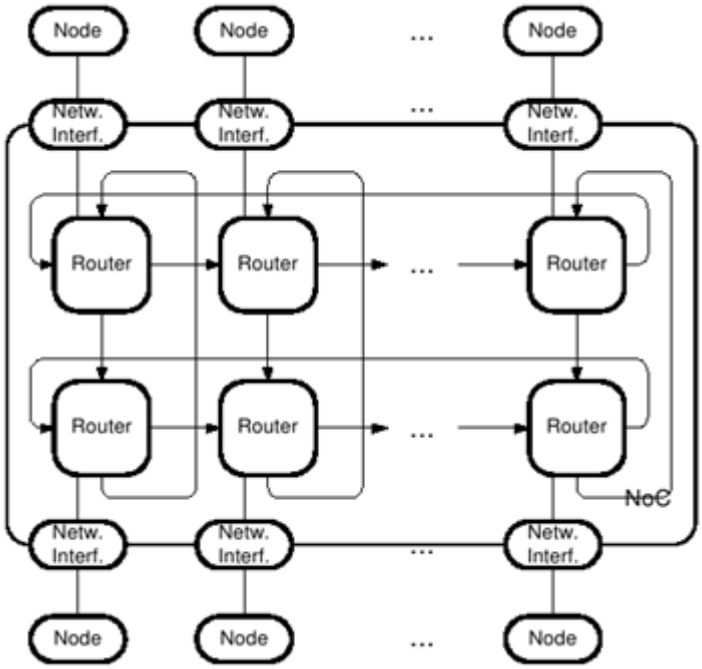


11 pav. TORUS komutatorių galimas adresacijos variantas [17]

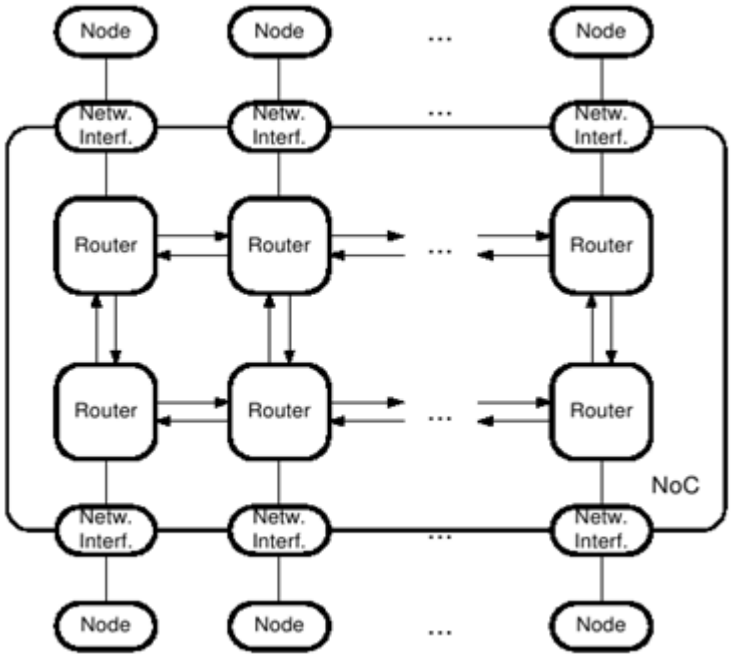


12 pav. Realiai realizuoto TORUS tinklo schema.

.



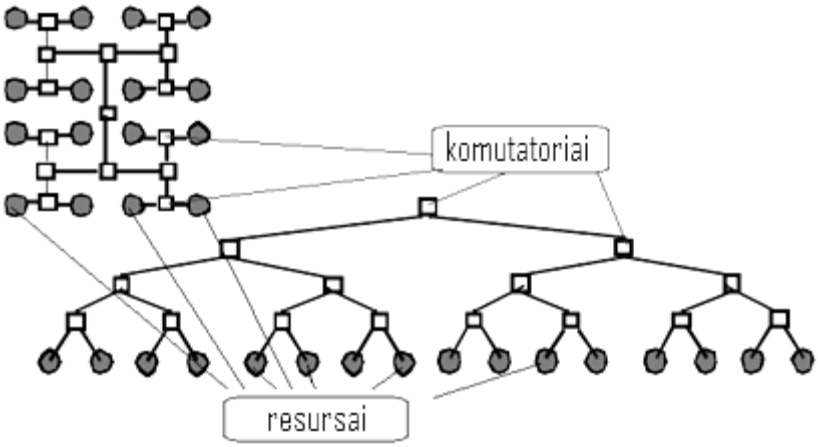
Torus tinko topologija yra labai panaši i MESH skirtumas tik tas kad yra naudojami papildomi laidai, dėl to skiriasi ir adresacija.Tai galima pamatyti esančiuose pavyzdžiuose



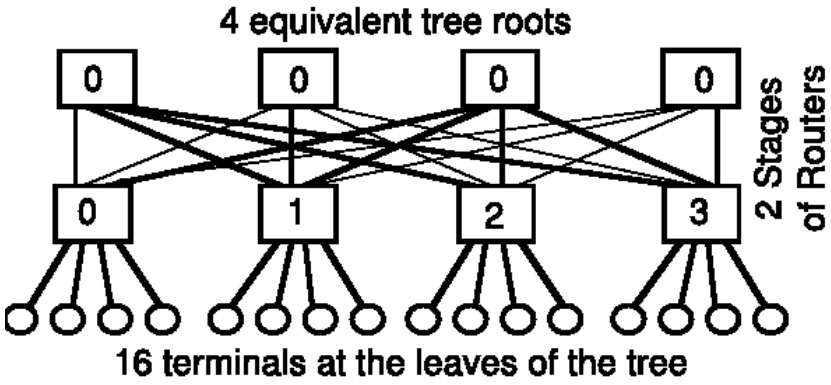
13 pav. TORUS ir MESH palyginimas

1.1.6. **SPIN**

SPIN – "A Scalable Programmable Interconnection Network". SPIN topologija naudoja medžio topologija, kuri pagrįsta komutavimo privalumais grupuojant atskiras tinklo dalis. SPIN struktūra tokia pat kaip fat-tree topologijos. Kaip teigia Leiserson [6] fat-tree yra topologija, kuri turi geriausia kainos ir efektyvumo santykį VLSI sistemose.



14 pav. SPIN NOC topologija yra fat-tree topologijos modifikacija.



15 pav. SPIN variacija. [11]

16 - ame paveikslėlyje pavaizduota fat-tree topologija su 16 lapų. Šie topologijos elementai lapai tai tinklo mazgai susisiekinėja per tinklo mazgus laikomus komutatoriais. Visi komutatoriai sujungti tarpusavyje dvikrypčiu ryšiu.

Fat-tree topologijoje yra dviejų rūšių komutatoriai: tie kurie turi tėvinius mazgus ir tie kurie neturi šių tėvininių mazgų. Jei visi komutatoriai su tėviniais mazgais sujungti su tokiu pat skaičiumi, vaikinius mazgus turinčiais komutatoriais, tai toks tinklas vadinamas non-blocking. Tada mazgų įėjimo galimybės yra tokios pat kaip ir išėjimo ir nepralaidumas ar kokie tai paketų užsikišimai jau nebus galimi tol kol kiekvienai įėjimo linijai bus viena išėjimo linija. [11]

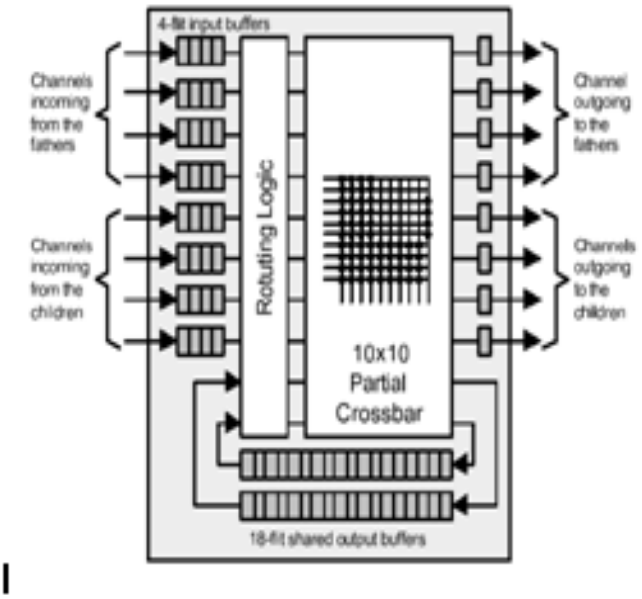
2 lentelė. Elementų kiekio proporcija fat-tree topologijoje. [11]

Mazgu skaičius	Komutatoriai	Jungtys
8	2	12
16	8	32
32	16	96
64	48	192
128	96	448

1.1.6.1 **SPIN komutavimas ir maršrutizavimas.**

SPIN siunčiami paketai gali varijuoti dydžiu. Prieš siuntimą paketai yra padalinami į atskirus paketus, kurie po to yra siunčiami tinkle. SPIN naudoja angliškai wormhole maršrutizavimą siunčiant flit'us per trumpiausią laiką.

Pirmiausia paketo dalis keliauja iš vaikinių mazgų į tėvinius mazgus. Po to paketas gali būti siunčiamas bet kuriuo neužimtu kanalu aukštyn medžiu.



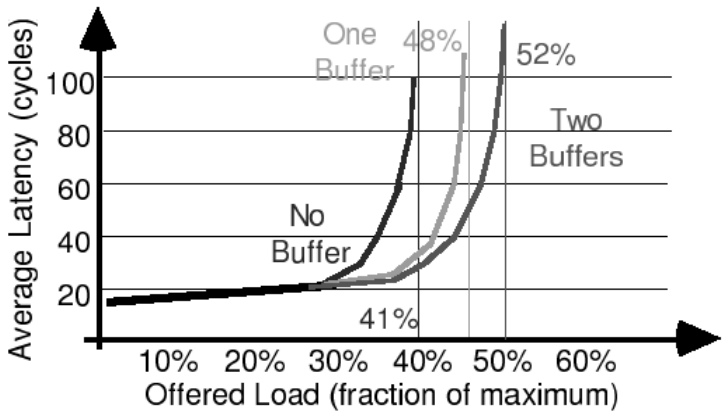
16 pav. SPIN maršrutizatorius.

SPIN komutatoriaus schematinis vaizdas pateiktas 17 paveikslėlyje. Matome pavaizduotus buferius ant kiekvienos įėjimo linijos iš tėvinių mazgų, hierarchiškai ir linijų įėjimai iš vaikų mazgų. Taip pat yra ir du „18 word“ išėjimo buferiai kuriais dalinasi išėjimo kanalai. Išėjimo buferiai naudojami perduodant duomenis iš įėjimų į išėjimus.

Pagrindinis SPIN komutatoriaus komponentas yra 10x10 komutatorius. Šis komponentas sprendžia ar paketai siunčiami žemyn medžiu ar užlaikomi išėjimo buferiuose jei siuntimo kanalai yra užimti.

1.1.6.2 **Darbo charakteristikos**

SPIN pesimistinis realizavimo įvertinimas buvo rastas simuliuojant su system-level simuliacijos įrankiu. Vienodos atsitiktinės distribucijos paketų tikslai buvo tariamai aplaidus visos informacijos lygyje. Be to mazgų grupavimas į medžio struktūrą ne visada galimas.



17 pav. SPIN naudingumo testo grafikas.

Paveikslėlyje 18 matome kad vidutinis flit paketo vėlinimas priklauso nuo SPIN apkrovimo su 32 terminalo mazgais ir su atitinkamu išėjimo buferių kiekiu kiekviename SPIN komutatoriuje. Paduotas apkrovimas yra ciklų rinkiniai kuriais paketai yra paduodami i buferius prijungtus prie kiekvieno mazgo terminale. Vėlavimai pateikti grafikuose parodo kiek laiko užtrunka buferiuose ir kiek laiko užtrunka paketo siuntimas tinklu.

Matome, kad buferio su dviem išėjimais maksimalus apkrovimas yra 50 %. Tai yra labai geras rodiklis kuris rodo, kad tinklo efektyvaus darbo apkrovimas gali būti intervale nuo 30% iki 80%. Kitos simuliacijos su trimis ar daugiau buferio išėjimų parodė tinklo strigimus ar darbo lėtėjimą.

1.1.7. Trumpas NOC architektūrų palyginimas

3 lentelė. Pateikiami architektūrų topologijų tarpusavio lyginamosios charakteristikos

Pavad.	Pagrindiniai pranašumai	Pagrindiniai trūkumai
MESH	1. Gerai valdomi elektrinai galios parametrai.	1. Ploto atžvilgiu reikalinga daug vietos komutatoriams.
TORUS	1. Tobulesnis už mech adresavimo būdas. 2. Pasitelkiant papildomas jungtis mažinami vėlavimai.	1. Reikalingi papildomi laidai todėl reikalingas papildomas plotas ir kai kada sukelia papildomus vėlavimus.
SPIN	1. Efektyviausiai naudojama VLSI (Very large-scale integration) sistemose	2. Kadangi Efektyviausia VLSI (Very large-scale integration)mažiau naudinga mažuose technologiniuose projektuose. 2. Palyginus sudėtingi komutatoriai

1.2 Gaminių linijos

Daug organizacijų naudoja programinių gaminių linijų kūrimo metodus, kad pagerinti kūrimo kokybę, pagreitinti išleidimo datą, bei pagerinti patį gaminį. Pagrindinis principas ir reikalavimas norint sukurti gaminių linijas yra gaminių linijų architektūros struktūra palaikanti gaminių variantiškumą. O variantiškumą išgaunamas iš programų atkartojimo metodų. [14]

1.2.1 Pagrindiniai gaminių linijų procesai.

18 pav. Pagrindiniai produktų linijų procesai [23]

- Projekto dalies kūrimas – tai procesas kurio tikslas sukurti sudedamąją produkto dalį.
- Produkto kūrimo procesas skirtas gaminti produktus susidedančius iš jį sudedančiųjų dalių.
- Projekto valdymo procesas skirtas techninių ir organizacinių problemų sprendimui. [23]

1.3 **Programų atkartojimas**

Pritaikius atkarojimo technologijų galimybes galima potencialiai pagerinti programų produktyvumą, sutrumpinti kūrimo procesą bei sumažinti projekto kainą. Dabartiniu metu konkrečios metodologijos kuri sujungtų atkartojimą, kaip standartini kūrimo etapą, ir faktorių, kurie veikia atkartojimą, reikšmingumo lygmuo yra kontraversiškas. Kita vertus kodo atkartojimas matuojamas objektiškai naudojant tinkamas priemones. [2]

1.3.1. **Atkartojimo metodai.**

Atkartojimo technologija yra komponentikos kontekstas. Atkartojimas pagal savo principus ir tikslus (galimybes) gali būti sudalintas į dvi dalis: intuityvus (ad hoc, oportunistinis) ir sisteminis. Be to pagal procesus (techniniai, netechniniai) atkartojimo metodologija dalinama į techninį ir netechninį (žmogiškasis faktorius standartizacijos, komponentų pirkimas-pardavimas).

Intuityvaus atkartojimo principai yra surasti, suprasti, modifikuoti ir naujai panaudoti atkartojamą objektą (specifikaciją, dokumentą, testų rinkinį, kodą, architektūrą, projektą).
Sisteminis atkartojimas yra orientuotas į panašių ar gimininingų sistemų šeimynos sukūrimą ir jo pagrindas yra srities inžinerija (srities analizė).

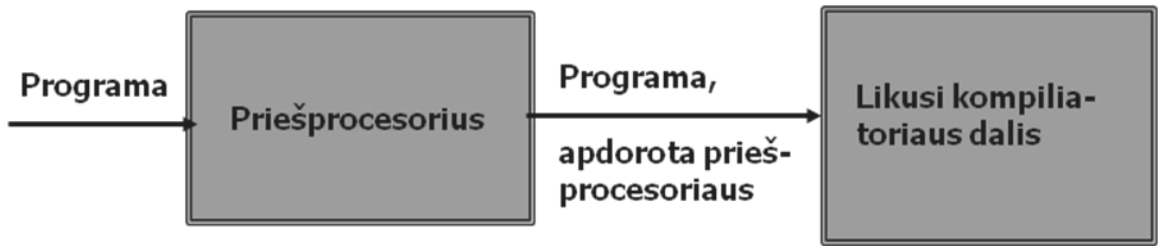
Atkartojimo metodus skirstomi pagal du kriterijus atvaizdavimo būdą (daliniai - kopijavimas, preprocesorinis apdorojimas ir bendri – aukštesnio lygio abstrakcijos, metaprogramavimas) ir atkartojimo principą (kompoziciniai metodai ir generavimu ar generatoriais grįsti metodai).

1.3.1.1 **Kodo generavimas**

Kodo generavimo metodai- generatoriais ir transformacinėmis sistemomis Generavimo metodo privalumai: leidžia pasiekti aukštesnį produktyvumą, atsikartojantys modeliai gali būti rūpestingai suprojektuoti patyrusio programuotojo, todėl yra aukštos kokybės Trūkumai: taikytini tam tikroje srityje, taikymas apribotas išeities kodui negali būti lengvai pritaikomas visose situacijose dažnai jie arba per daug bendri, arba per daug specifiniai.

1.3.1.2 **Preprocesorinis apdorojimas.**

Jei sistemos kodas yra aprašytas sysmemc programavimo kalba, o šio tyrimo metu taip ir yra, galima įgyvendinti preprocesorinį atkartojimo metodą. Preprocesoriniui apdorojimui naudojamos priemones yra kompiliatoriaus direktyvos, kuriomis galima pasinaudoti norint pasirenkanį kompiliuojamo kodo variacijas.



19 pavaikslėlis. Preprocesoriaus veikimo schema.

Priešprocesorius - tai integruota C/C++, systemc, taip pat kompiliatoriaus dalis, kuri atlieka paruošiamuosius veiksmus, prieš pradedant kompiliuoti programą. Priešprocesorius valdomas direktyvomis, t.y. komandomis, kurios nurodo, kokie veiksmai turi būti atlikti prieš pradedamos programos kompiliavimą. Tai butų: #define, #undef, ,#ifndef, #ifdef, #endif, #else, #elif , #include, #error. Direktyvos apdorojamos pirminiame programos kompiliavimo etape, o jų naudojimas apibrėžtas C standarte.

1.3.2. **Atkartojimas vienlusčių tinkluose.**

Vienlusčių technologijos kūrimo metodologija yra padalijamojo tipo programų kūrimo alternatyva. Pagrindiniai tokio kūrimo, panaudojus atkarojimą, privalumai yra:

- Padidėjęs duomenų siuntimo pralaidumas keičiant tinklo tipą;
- Leidžia planuoti standartinį laidų išdėstymą;
- Galima įdiegti maršrutizatorių atkartojimo metodus;
- Galimas mažesnis energijos sunaudojimas planuojant trumpesnius laidus tarp tinklo resursų;[3]

Vienlusčių tinklas turi būti pritaikytas atskirai kiekvienam specifiniam tinklo panaudojimo atvejui, pavyzdžiui jei reikalingi mažo talpumo paketų siuntimui naudojamų

atminties registrų reikalavimas ar specifinė vienlusčių tinklo topologija. Esant tokiems įvairiems reikalavimams yra labai naudingas greitas tokio vienlusčių tinklo modifikavimas parametrizavimas. Kad lengviau sukurti ir įdiegti tokį bendrini vienlusčių tinklų komponentą yra naudojama kodo atkartojimo technologijos. Toks komponentas galėtu būti bendriniu su reikiamais atributais:

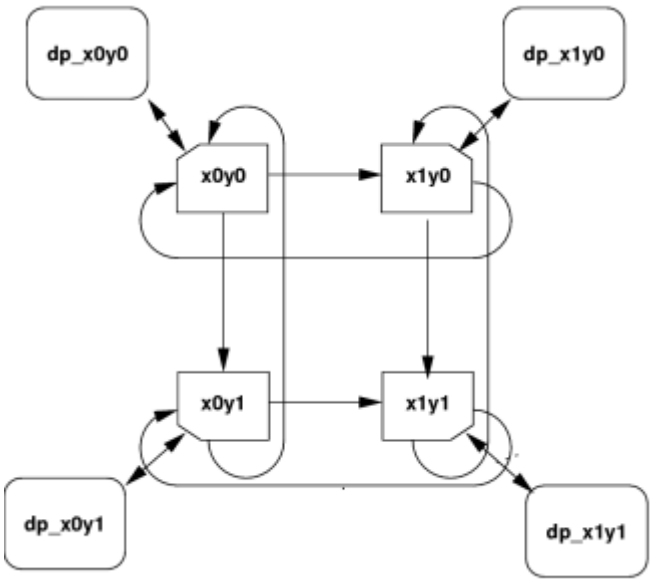
- Savarankiškumas;
- Identifikavimas**;**
- **Funkcionalumas*;***
- **Sąsajos***;*
- Dokumentacija;
- Statusas;

2. TYRIMŲ DALIS

2.1 **Realizavimo variantų analizė**

2.1.1 **Tiriamasis vienlusčių tinklas**

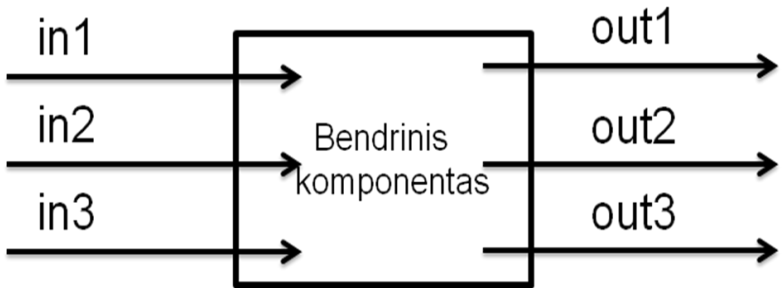
Tinklas jungiantis keturis mMIPS procesorius yra *torus* topologijos tipo su E-cube maršrutizavimo variantu. Tai yra kiekvienas paketas tinklu pirmiausia keliauja x kryptim, apačioje pavaizduotos schemos atveju būtu horizontaliai, kol pasiekia norima maršrutizatorių, kurį nustato pagal unikalų x adresą Po to paketas pasiunčiamas y kryptimi, vertikaliai, tol kol pasiekiamas maršrutizatorius pagal y maršrutizatoriaus adresą. Kadangi maršrutizatoriai sujungti tinklu, kur paketai gali keliauti tik viena kryptimi nukeliavus i tinklo pabaigą jis nukreipimas i tinklo pradžią, jei reikalinga. Tinklo elementai sujungti 18 bitų, laidų platumo juostomis (16 bitų skirti duomenims siųsti , ir 2 bitai kontrolės tikslams). MIPS mikro procesoriai prijungti prie maršrutizatorių dvikrypčiu ryšiu, o maršrutizatoriai tarpusavyje tik vienos krypties ryšiu. Komunikavimas šiuo vienlusčių tinklu yra vykdomas rankų paspaudimo arba *wormhole* sinchroniniu darbo principu.



20 pav. Tiriamo vienlusčių tinklo schema, 4 maršrutizatoriai (xYyY), ir keturi miniMips procesoriai (dp_xYyY) [22]

2.1.2 **Bendrinis komponentas, maršrutizatorius.**

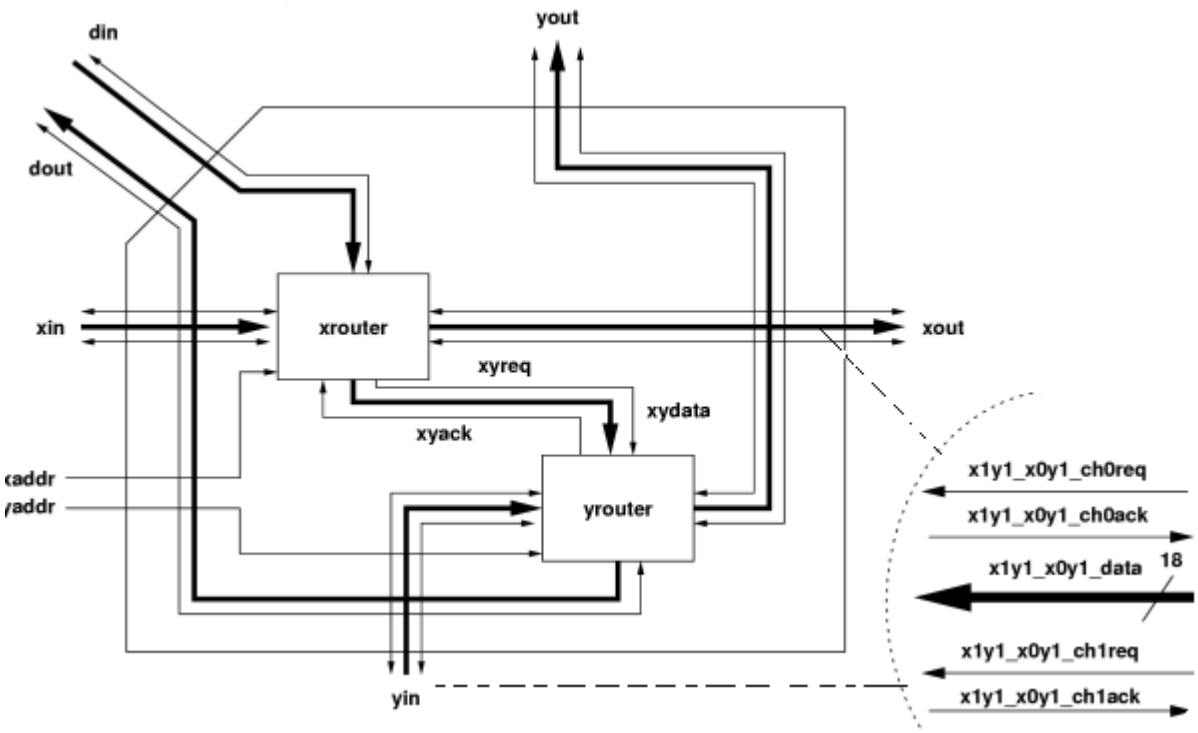
Naudojant atkartojimo preprocesorinį metodą, sukurti bendrinį vienlusčių tinklų komponentas, maršrutizatorius.



21 pav.. Bendrinis komponentas, maršrutizatorius, trys įėjimai in1, in2 ir in3, ir trys išėjimai out1, out2, out3.

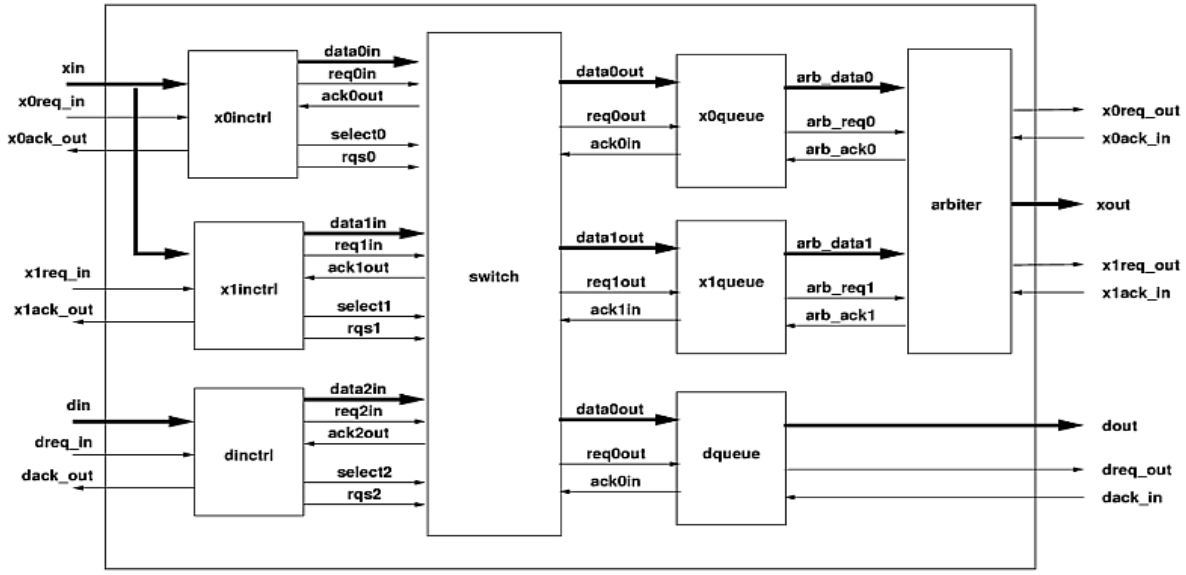
2.1.2.1 **Maršrutizatorius nr. 1, maršrutizuojant paketus virtualiais perdavimo kanalais.**

Vienas iš deadlock-free gavimo variantų yra naudoti virtualius paketų persiuntimo kanalus, konkrečiai šiuo atveju yra du virtualus kanalai 0 ir 1. Tam kad tai įgyvendinti torus sistemos maršrutizatoriui reikalingi papildomi elementai. Dviejų dimensijų tinkle maršrutizatorių galima realizuoti kaip dviejų atskirų maršrutizatorių, kurie duomenis gali siusti viena kryptimi, kombinaciją, 22 paveikslas.



22 pav. Maršrutizatorius susidedantis iš dviejų paprastesniu maršrutizatorių: xrouter ir yrouter.[22]

Matome kad maršrutizatorius galintis maršrutizuoti paketus x ir y kryptimis susideda iš dviejų paprastesnių maršrutizatorių kurie maršrutizuoja paketus tik viena kryptimi, x arba y, matome 23 paveiksle.

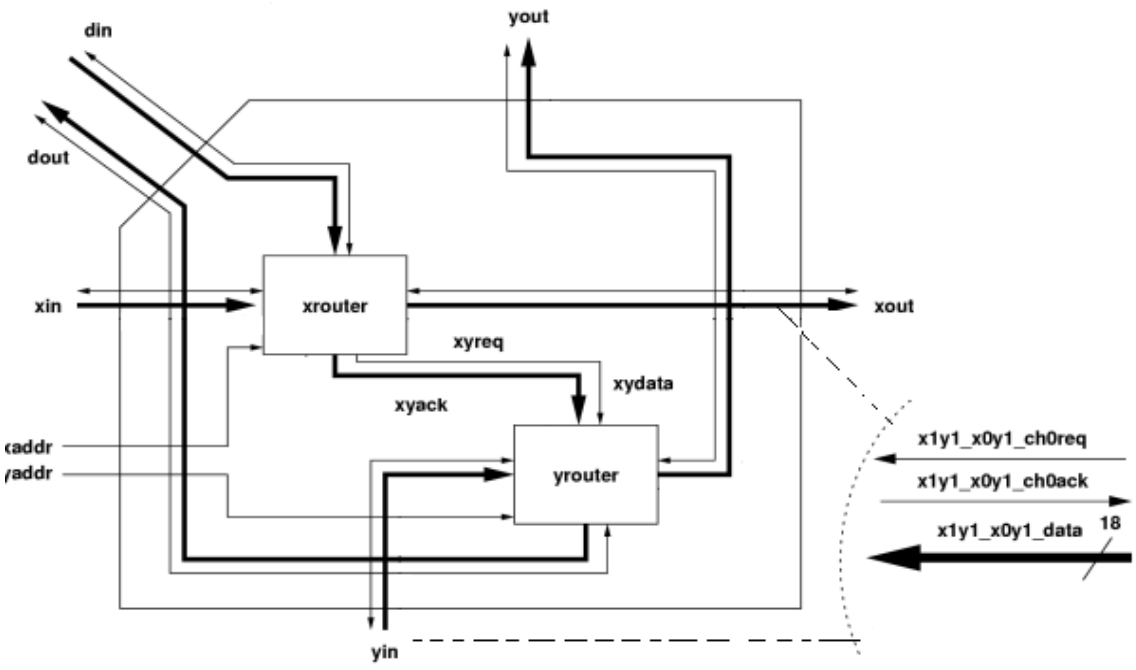


23 pav. Paprastesnės sudėties maršrutizatorius, galintis maršrutizuoti tik viena kryptimi.[22]. x0inctrl,

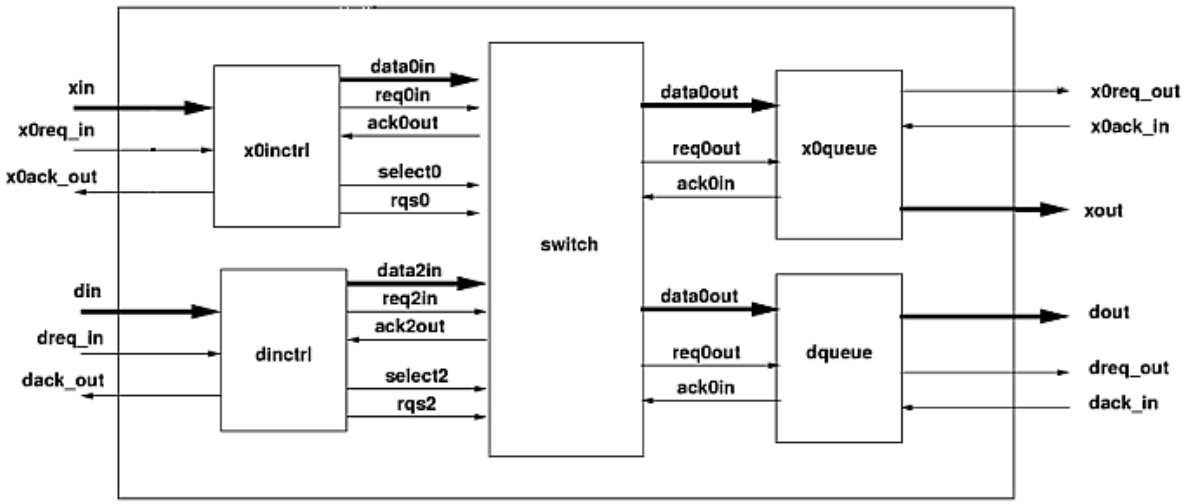
x1inctrl, dinctrl– įėjimo kontroleriai; switch – komutatorius, x0queue, x1queue, dqueue – išėjimų buferiai; arbitre – virtualių kanalų išėjimų arbitras.

Šis vienos dimensijos maršrutizatorius susideda iš trijų identiškų įėjimo kontrolerių, iš kurių du yra skirti virtualių kanalų duomenų persiuntimui, matome, kad virtualūs duomenų perdavimo kanalai dalinasi tuo pačiu fiziniu įėjimu, bet turi atskirus įėjimo kontrolerius, Maršrutizatorius, nenaudojant virtualių perdavimo kanalų. Gavęs užklausą įėjimo kontroleris parsisiunčia ir ištirią gautus duomenis. Jei paketas yra pirmas transakcijos, kontroleris prašo kad komutatorius sujungtu su reikiamu išėjimo buferiu. Buferio pasirinkimas priklauso nuo pradinio paketo siunčiamo adreso. Kai sujungimas pavyko, nustatomas kelias pro įėjimo kontrolerį, komutatorių iki išėjimo buferio, ir duomenys siunčiami tol kol buferis būna pilnas. Kai virtualiam kanalui reikia priėjimo prie fizinio kanalo jis kreipiasi i išėjimo arbitrą, kuris nustato ar maršrutizatoriaus išėjimas užimtas.

2.1.2.2 Maršrutizatorius nr.2 - nenaudojant virtualių kanalų



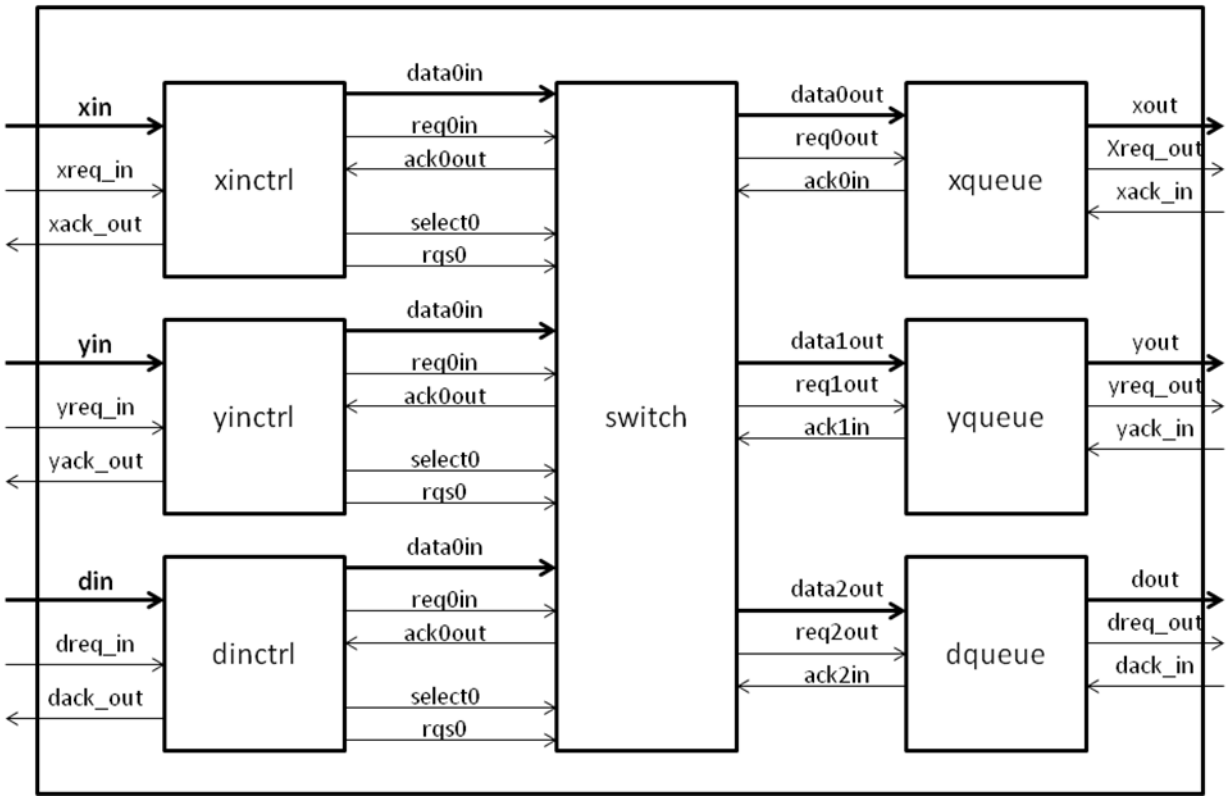
24 pav. Maršrutizatorius susidedantis iš dviejų paprastesniu maršrutizatorių xrouter, yrouter , kuris nenaudoja virtualių 0 ir 1 duomenų kanalų.



25 pav. Maršrutizatorius susidedantis iš dviejų paprastesniu maršrutizatorius nenaudojant virtualių 0 ir 1 duomenų kanalų. x0inctrl, dinctrl– įėjimo kontroleriai; switch – komutatorius, x0queue, dqueue – išėjimų buferiai;

2.1.2.3 Maršrutizatorius nr. 3 - naudojant tik vieną komutatorių.

Šis maršrutizatorius nesusideda iš atskirų dviejų paprastesnės logikos maršrutizatorių skirtingai nei prieš tai apibūdinti maršrutizatoriai. Tokį maršrutizatoriaus supaprastėjimą įtakojo sudėtingesnes logikos įėjimo kontroleris, kuris sugeba atpažinti viena iš trijų, xout, yout, ar dout paketo keliavimo krypčių.



26 pav.Maršrutizatorius turintis tik viena komutatorių, bet sudėtingesnės logikos įėjimo kontrolerį. xinctrl, yinctrl, dinctrl– įėjimo kontroleriai; switch – komutatorius, xqueue, yqueue, dqueue – išėjimų buferiai.

Kiekvienas maršrutizatoriaus įėjimas, išėjimas sujuntas su fiziniais tinklo kanalais. Tokai maršrutizatoriaus sudėtis turėtų žymiai sumažinti sudedamąjį maršrutizatoriaus ventilių skaičių, ypač lyginant su pirmuoju maršrutizatoriaus variantu.

2.1.2.4 Maršrutizatoriaus siunčiamų duomenų paketo, išėjimo buferio, magistralės dydis.

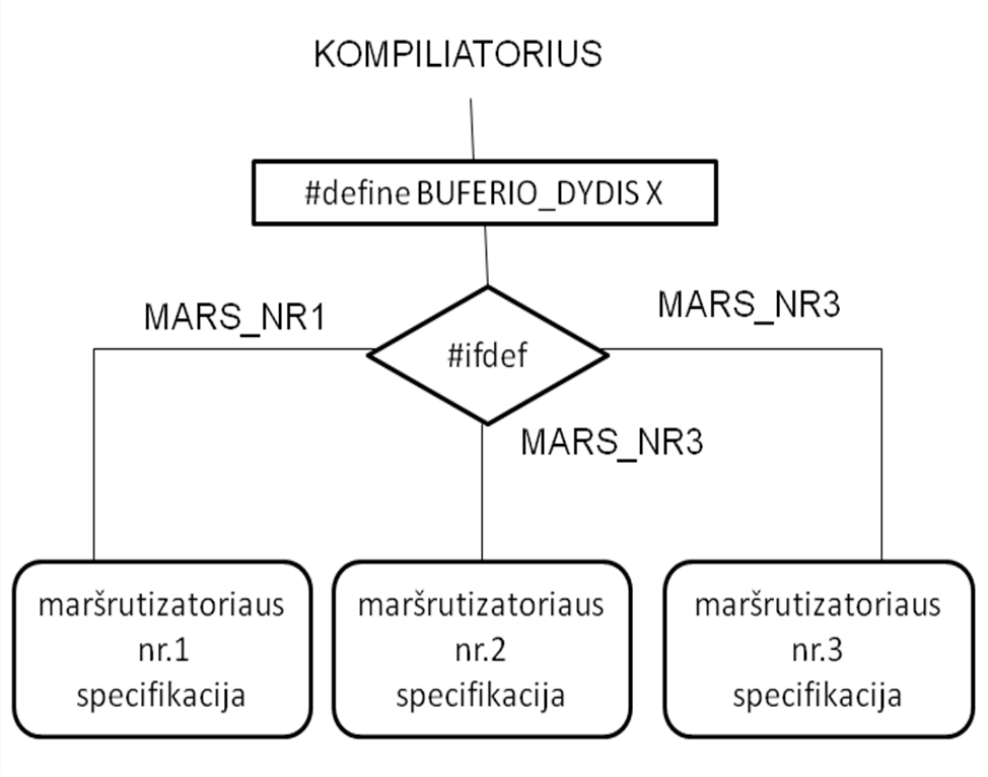
Duomenys vienlusčių tinklu yra siunčiami nustatyto dydžio komponentų parametrais, kurie tiesiogiai vienas nuo kito priklauso:

- Siunčiamo paketo dydis.
- Magistralių jungiančių maršrutizatorius ir tinklo resursus plotis.
- Maršrutizatoriaus išėjimo eilės buferio dydis.

Kad būtų galima, patestuoti kaip tinklo savybės priklauso nuo šio parametro. įdiegtas kintamasis, taip pat preprocesoriaus pagalba, leidžiantis nustatyti šį dydį. Tam kad visa bendra sistema, vienlusčių tinklas ir į jį įjungti miprocesoriai

2.2 Struktūros ir veikimo principo pagrindimas. Veikimo algoritmo sudarymas.

Vienlusčių tinklas aprašytas systemc kodavimo kalba. Todėl galimas preprocesorinis atkartojimo metodas. Pasinaudojus procesoriaus direktyvomis galima atrinkti ir idiegti i tinklą norimą maršrutizatoriaus variantą. Žemiau pavaizduotame, 28 paveiksle, matome kaip kompiliatoriaus pagalba, atrenkamas maršrutizatoriaus egzempliorius, kuris bus naudojamas vienlusčių tinkle.



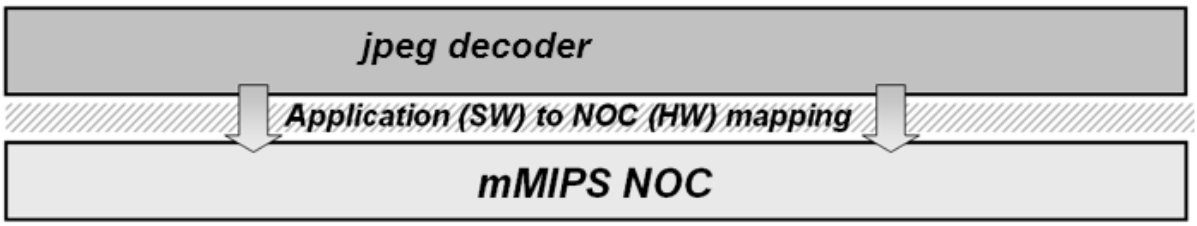
27 pav. Egzempliorių pasirinkimo algoritmas. MARS_NR1, MARS_NR2, MARS_NR3 –preprocesoriaus apibūdinti kintamieji.

Dydis X 28 paveiksle nusako:

- Įėjimo kontrolerio buferio dydį
- Išėjimo buferio dydį
- Duomenims siųsti skirtų magistralių plotį

2.3 **Jpeg dekoderis.**

Vienas iš geriausių testavimo metodų yra realus programiškas vienlusčių tinklo išbandymas. Toks vienlusčių tinklas su keturiais MIPS procesoriais gali būti pritaikytas jpeg dekodavimui, padalinant darbus, kiekvienam procesoriui atskirai. Jpeg dekoderis dekoduoja jpeg paveiksliuką į bitmap paveiksliuką, naudojantis dekodavimo metodu vadinamu „basline decoding process“. Kiekvienas vienlusčių tinklo variantas, su atskirais maršrutizatoriaus egzemplioriais, įtakoja visos sistemos bendrą dekodavimo proceso atlikimo laiką, todėl vienlusčių tinklus nuspręsta patestuoti ir simuliuojant ši dekodavimą.



28 pav. jpeg dekodavimo algoritmų įgyvendinimas i vienlusčių tinklą

Plačiau apie šį bandymą ir galima paskaityti [21] literatūros šaltinyje.

3. REZULTATAI

3.1 **Sintezės rezultatai.**

Turint vienlusčių tinklą su bendriniu komponentu, kurio egzemplioriais galime varijuoti gauname sintezuojamus vienlusčių tinklų aprašus. Žemiau, 4 lentelėje, pavaizduoti rezultatai varijuojant visų maršrutizatorių tipais bei dvejomis 18bit ir 34bit (du bitai skirti kontroliės tikslams, kad nustatyti paketo paskirtį: pradinis paketas, duomenų paketas, transakcijos baigiamasis paketas), magistralės, išėjimo buferio,parametrų reikšmėmis.

4 lentelė. Sintezės rezultatai ploto prioritetas prieš vėlinimą.

Bandymo Numeris.	Ventilių sk.	Laidų sk.	Galios suvartojimas (uW)	Plotas (nm)	Vėlinimas (ns)
	Optimizuota ploto atžvilgiu, 18bitų magistralė, ir buferiai.				
1 (marš. Nr1)	1788	2105	2.6594)	99542.9453	-9.61
2 (marš. Nr2)	1136	1395	2.6207	61677.543	-9.08
3 (marš. Nr3)	981	1240	2.6115	52958.5273	-9.64
	Optimizuota vėlinimų atžvilgiu, 18bitų magistralė, ir buferiai.				
4 (marš. Nr1)	1815	2132	2.6581	99398.5781	-9.61
5 (marš. Nr2)	1152	1411	2.6198	61535.0313	-8.78
6 (marš. Nr3)	978	1237	2.6077	52782.4141	-9.5
	Optimizuota ploto atžvilgiu, 34 bitų magistralė, ir buferiai.				
7 (marš. Nr1)	2219	2776	2.7022	129849	-9.64
8 (marš. Nr2)	1446	1913	2.6533	79534.7	-9.67
9 (marš. Nr3)	1228	1695	2.644	68325.9	-9.65
	Optimizuota vėlinimų atžvilgiu, 34bitų magistralė, ir buferiai.				
10 (marš. Nr1)	2250	2807	2.7011	130077	-9.67
11 (marš. Nr2)	1464	1931	2.6543	79564.2	-9.07
12 (marš. Nr3)	1202	1669	2.64	67803.6	-9.61

marš.Nr.1 – maršrutizatorius turintis du virtualius kanalus duomenims perduoti.
marš.Nr.2 – maršrutizatorius neturintis virtualių kanalų duomenims perduoti.
marš.Nr.3 – maršrutizatorius kurio sudėtyje tik vienas komutatorius.
Ventilių, laidų bei prievadų skaičius lentelėje pateiktas vienetais. Plotas pateiktas technologijos nustatytais ploto vienetais. Vėlinimais taktais, prilygstančiais nano sekundėms. Modelio suvartojama energija matuojama mW. Schema maitinama 5V įtampa

Sintezės bandymų variacija 4 lentelėje:

1) Sintezės rezultatas gautas optimizuojant rezultatą **ploto** atžvilgiu. Tinkle naudojamas maršrutizatorius Nr.1,naudojantis virtualius duomenų perdavimo kanalų. Jungimosi magistralės bei maršrutizatoriaus išėjimų buferiai 18 bitų.

2) Sintezės rezultatas gautas optimizuojant rezultatą **ploto** atžvilgiu. Tinkle naudojamas maršrutizatorius Nr.2, nenaudojantis virtualių duomenų perdavimo kanalų. Jungimosi magistralės bei maršrutizatoriaus išėjimų buferiai 18 bitų.

3) Sintezės rezultatas gautas optimizuojant rezultatą **ploto** atžvilgiu. Tinkle naudojamas maršrutizatorius Nr.3. Jungimosi magistralės bei maršrutizatoriaus išėjimų buferiai 18 bitų.

4) Sintezės rezultatas gautas optimizuojant rezultatą **vėlinimo** atžvilgiu. Tinkle naudojamas maršrutizatorius Nr.1,naudojantis virtualius duomenų perdavimo kanalų. Jungimosi magistralės bei maršrutizatoriaus išėjimų buferiai 18 bitų.

5) Sintezės rezultatas gautas optimizuojant rezultatą **vėlinimo** atžvilgiu. Tinkle naudojamas maršrutizatorius Nr.2, nenaudojantis virtualių duomenų perdavimo kanalų. Jungimosi magistralės bei maršrutizatoriaus išėjimų buferiai 18 bitų.

6) Sintezės rezultatas gautas optimizuojant rezultatą **vėlinimo** atžvilgiu. Tinkle naudojamas maršrutizatorius Nr.3. Jungimosi magistralės bei maršrutizatoriaus išėjimų buferiai 18 bitų.

7) . Sintezės rezultatas gautas optimizuojant rezultatą **ploto** atžvilgiu. Tinkle naudojamas maršrutizatorius Nr.1,naudojantis virtualius duomenų perdavimo kanalų. Jungimosi magistralės bei maršrutizatoriaus išėjimų buferiai 18 bitų.

8) Sintezės rezultatas gautas optimizuojant rezultatą **ploto** atžvilgiu. Tinkle naudojamas maršrutizatorius Nr.2, nenaudojantis virtualių duomenų perdavimo kanalų. Jungimosi magistralės bei maršrutizatoriaus išėjimų buferiai 34 bitų.

9) Sintezės rezultatas gautas optimizuojant rezultatą **ploto** atžvilgiu. Tinkle naudojamas maršrutizatorius Nr.3. Jungimosi magistralės bei maršrutizatoriaus išėjimų buferiai 18 bitų.

10) Sintezės rezultatas gautas optimizuojant rezultatą **vėlinimo** atžvilgiu. Tinkle naudojamas maršrutizatorius Nr.1,naudojantis virtualius duomenų perdavimo kanalų. Jungimosi magistralės bei maršrutizatoriaus išėjimų buferiai 34 bitų.

11) Sintezės rezultatas gautas optimizuojant rezultatą **vėlinimo** atžvilgiu. Tinkle naudojamas maršrutizatorius Nr.2, nenaudojantis virtualių duomenų perdavimo kanalų. Jungimosi magistralės bei maršrutizatoriaus išėjimų buferiai 34 bitų.

12) Sintezės rezultatas gautas optimizuojant rezultatą **vėlinimo** atžvilgiu. Tinkle naudojamas maršrutizatorius Nr.3. Jungimosi magistralės bei maršrutizatoriaus išėjimų buferiai 34 bitų.

3.2 **Jpeg dekodavimo simuliacijos rezultatai**

Jpeg dekoderis sistemoje simuliuoti paleistas standartiniais nustatymais. Dekoduoti 32x24 taškų spalvotą paveiksliuką simuliatorius trunka 29 valandas Pentium III 1GHz GNU/Linux 2.4.20 su 2048 MB [21] .Šis procesas trunka 604 milisekundžių realiame FPGA produkte. [21] Testavimo rezultatai atlikti Pentium Core Duo 1.8 GNU/Linux 2.6.21 su 1024MB darbinės atminties, pavaizduoti žemiau esančiame grafike.

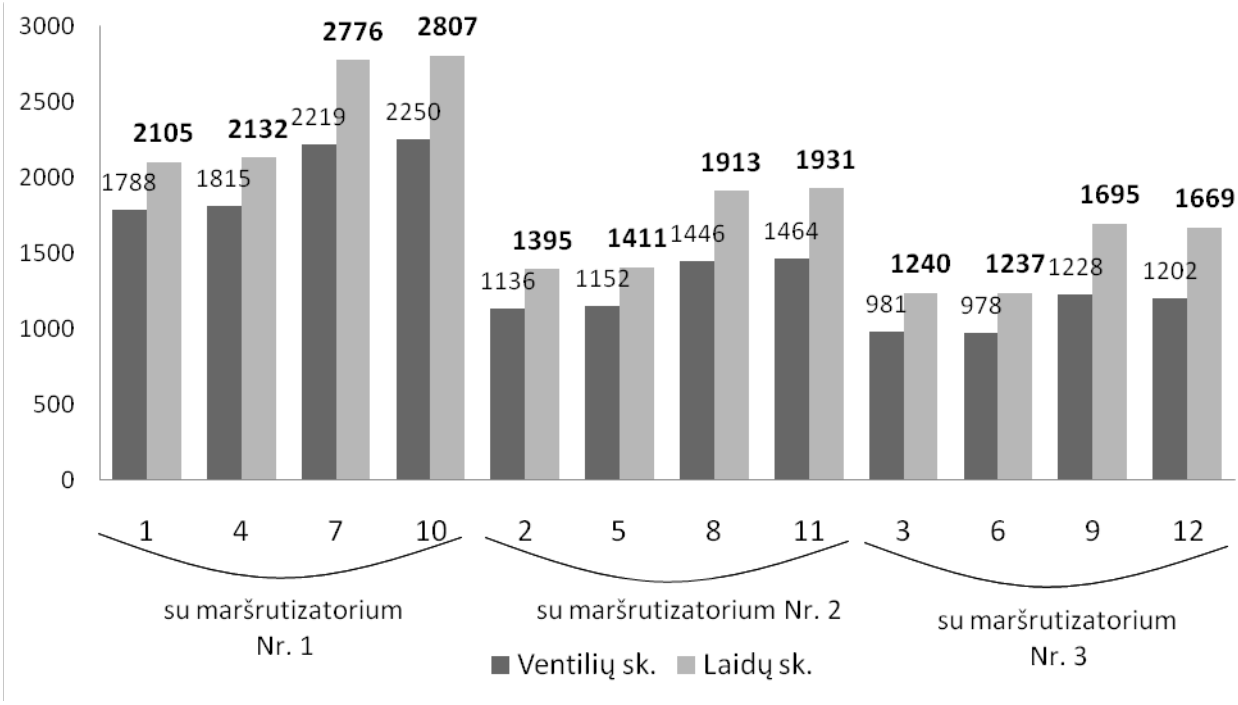
5 Lentelė.Procesorių sunejungtų tinklu, paveiksliuko dekodavimo darbo laikas.

Maršrutizatorius	Laikas
Nr. 1	10 val. 6min
Nr. 3	9 val. 53 min
Nr. 3	8 val. 30 min

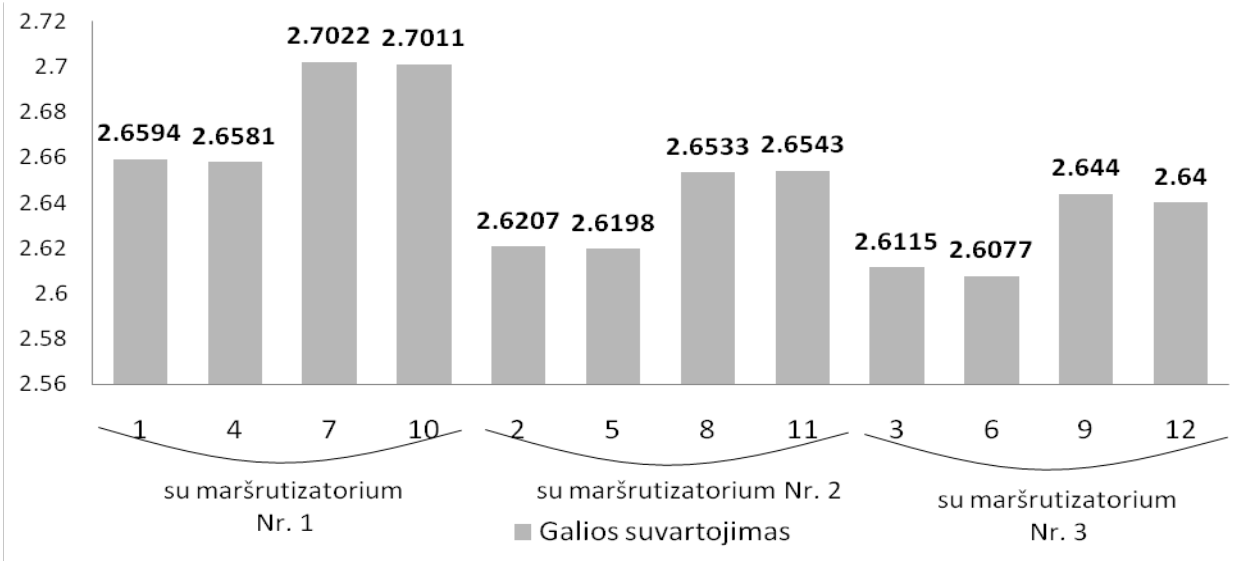
Lentelėje pateikti duomenys tik su 18bit platumo magistralėmis ir išėjimo buferiais, nes norint pakeisti vienlusčių tinklo veikimą kartu su procesoriais, iš 18 bitų į 34 bitus, reikia keisti ir kai kurias, mikroprocesorių savybes, o tai reikalauja papildomo tyrimo ir laiko.

4. DISKUSIJA

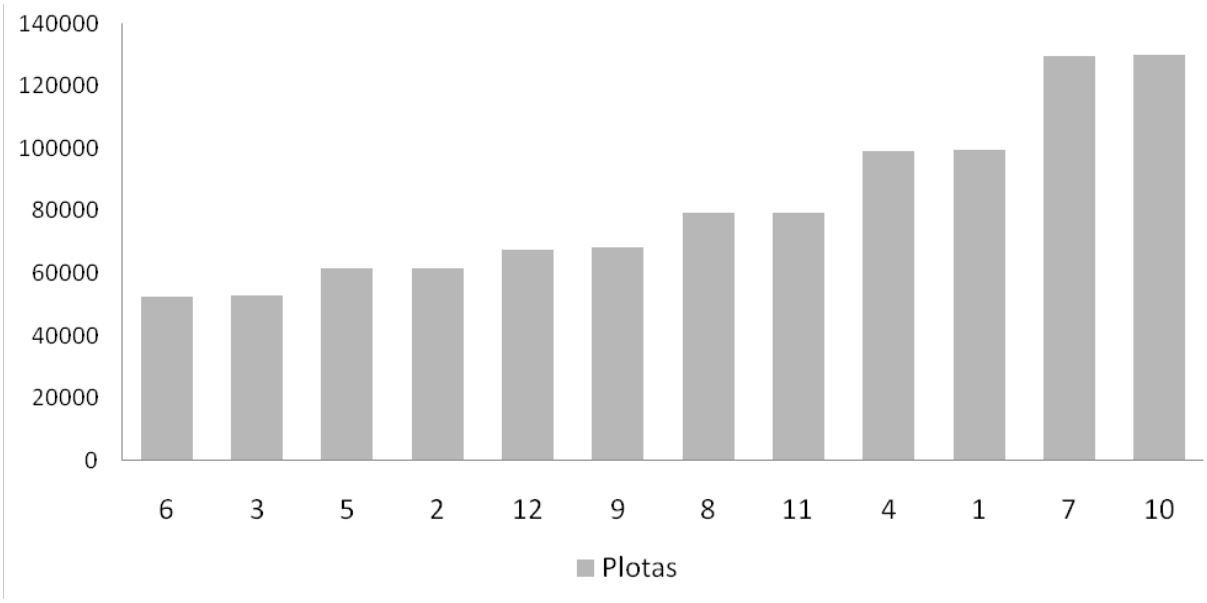
4.1 **Sintezės rezultatų analizė**



29 pav. Bandymų sintezės rezultatų ventilių ir laidų skaičius. X ašis nusako bandymo nr.

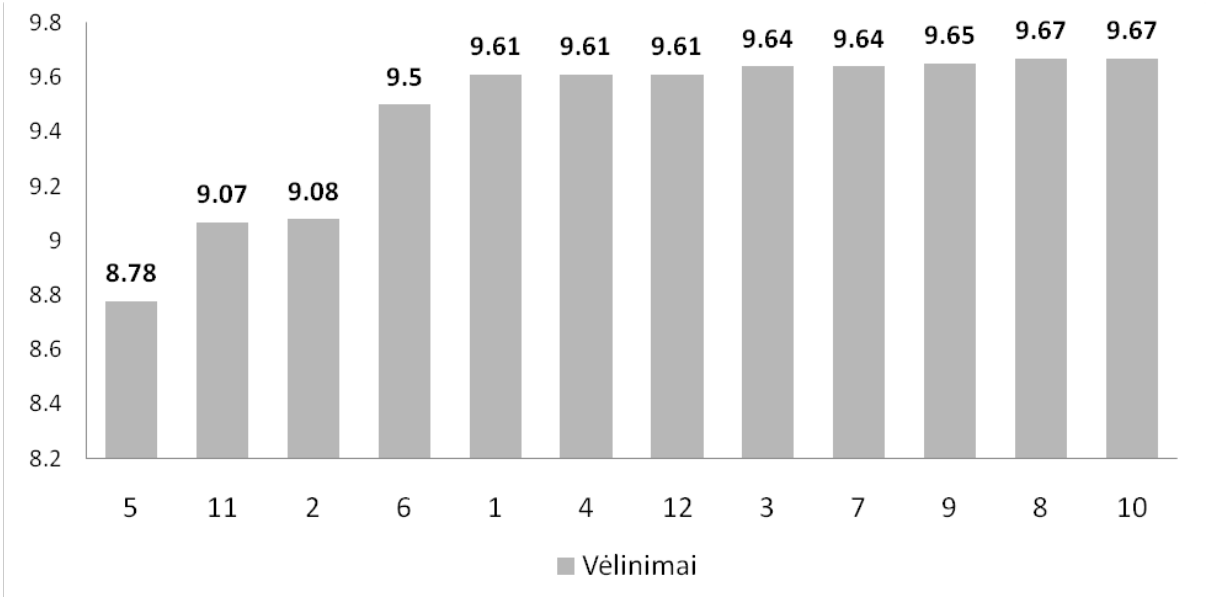


30 pav. Bandymų sintezės rezultatų galios parametrai. X ašis nusako bandymo nr.



31 pav. Bandymų sintezės rezultatų ploto parametrai. X ašis nusako bandymo nr.

Mažiausias plotas gautas, sintezės metu optimizuojant rezultatą vėlinimo atžvilgiu. Tinkle naudojamas maršrutizatorius Nr.3. Jungimosi magistralės bei maršrutizatoriaus išėjimų buferiai 18 bitų.

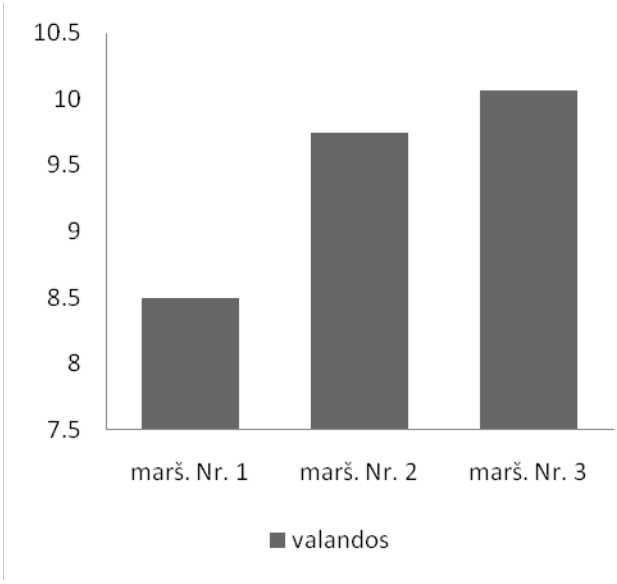


32 pav. Bandymų sintezės rezultatų ploto parametrai. X ašis nusako bandymo nr.

Mažiausias vėlinimas gautas optimizuojant susintezuotą rezultatą vėlinimo atžvilgiu. Tinkle naudojamas maršrutizatorius Nr.2, nenaudojantis virtualių duomenų perdavimo kanalų. Jungimosi magistralės bei maršrutizatoriaus išėjimų buferiai 18 bitų.

Matome, kad geriausios charakteristikos priklauso vienlusčių tinklui su maršrutizatoriumi nr. 3. O prasčiausios su maršrutizatoriumi nr.1, taigi dead-lock įgyvendinimas „kainuoja“ nemažai.

4.2 **Jpeg dekodavimo simuliacijos laikas.**



33 pav. Jpeg dekoderio simuliacijos laikų palyginimas valandomis. marš.Nr.1 – maršrutizatorius turintis du virtualius kanalus duomenims perduoti. marš.Nr.2 – maršrutizatorius neturintis virtualių kanalų duomenims perduoti. marš.Nr.3 – maršrutizatorius kurio sudėtyje tik vienas komutatorius.

Pastebėtas žymesnis simuliacijos darbo laiko sutrumpėjimas naudojant maršrutizatorių nr. 3. Išvada būtų tokia kad sudėtinis maršrutizatorius iš dviejų sudedamųjų paprastesnių maršrutizatorių, galinčių maršrutizuoti viena kryptimi, priverčia paketą praeiti per du išėjimo buferius, o tai reikalauja papildomo laiko.

IŠVADOS

Darbe pritaikyta atkartojimo metodika testuoti vienlusčių tinklų architektūros savybėms. Keletas atkartojimo metodų buvo aptarta ir vienas įgyvendintas. Tyrimas atskleidžia, kad atskyrus iš jau projekto atkartojamą komponentą galimą jį efektyviai panaudoti bendrinio komponento kūrimui, ir vėliau sistemos konfigūravimui ir svarbiausia testavimui. Kuo kodas labiau adaptuotas atkartojimui - tuo didesni tokio kodo privalumai, kurie leidžia geriau suprasti ir testuoti sistemas tokias kaip vienlusčių tinklai. Šiuo atveju sukurtas vienlusčių tinklo bendrinis maršrutizatoriaus komponentas, leidžiantis varijuoti keletu vienlusčių tinklo parametrų. Kuriant vienlusčių tinklus atkartojimo technologijos pagreitina testavimo laiką, taip pagerina tokių sistemų analizę, pagal kurią galima daryti reikalingas išvadas sprendžiant tokių sistemų gaminimo uždavinius.

ŠALTINIAI IR LITERATŪRA

[1] B. Towles W. Dally, *Route packets, not wires: On-chip interconnection networks*.

[2] Counsell S., Youssef Hassoun, Roger Johnson, *Code Reuse Through Reflection: An Empirical Perspective*.

[3] Chan Jeremy, Sri Parameswaran *NoCGEN A Template Based Reuse Methodology for Networks on Chip Architecture.*

[4] Carrabina Jordi, Antoni Portero, Ramon Pla *SystemC Implementation of a NoC*.

[5] C. Zeferino, M. Kreutz, L. Carro, A. Susin, 2002 *A study on Communication Issues For Systems-on-Chip,* 121-126.

[6] C. Leiserson, 1985 *Fat-trees: Universal networks for hardware-eficient supercomput- in.*

[7] Daniel Wiklund, 2005 „Development and Performance Evaluation of Networks on Chip“

[8] Krueger W., 2004 *Software Product Line Reuse in Practice Charles*.

[9] K. Goossens 2005 *Ethereal Network on Chip: Concepts, Architectures, and Implementations.*

[10] Micheli G. D., L. Benini, , 2002 *Networks on Chips a NewSOC Paradigm.*

[11] Michael Kochte, 2003 Seminar: Networks on Chip, *Packet-Switching Networks on Systems-on-a-Chip*

[12] Marcelo Lubaszewski Luigi Carroll, Erika Cota, Fliivio Wagner, *Powe-aware NoC Reuse on the Testing of Core-based Ssystems.*

[13] Muhammad Ali, Michael Welzl, Martin Zwicknagl *Networks on Chips: Scalable Interconnects for Future Systems on Chips;* Institute of Computer Science, university of Innsbruck, Austria.

[14] Mark Staples National ICT Australian Technology Park; Derrick Hill Dialect Solutions, *Experiences Adopting Software Product Line Development without a Product Line Architecture.*

[15] Ren´e van Leuken, 2005 *Asynchronous Network-on-Chip Architecture Performance Analysis MSC Thesis*.

[16] Cristian Grecu, Michael Jones 2005 *Performance Evaluation and Design Trade-Offs for Network-on-Chip Interconnect Architectures*.

[17] Stefano Santi, Bill Lin, Ljupco Kocarev, Gian Mario Maggio, Riccardo Rovatti, Gianluca Setti *On the impact of traffic statistics on quality of service for network on chip.*

[18] Zhang Hui, Marlene Wan, Varghese George, and Jan Rabaey *Interconnect architecture exploration for low energy reconfgurable single-chip DSPs.*

[19] Zeferino, C.A., Cota, E.F.; Kreutz, M.E., Carro, L., Lubasrewski, M. and Susin, A.A. *The Impact of NoC Reuse on the Testing of Core-based Systems,* in IEEE VLSl Tesf Symposium 2003.

[20] http://cse.stanford.edu/class/sophomore-college/projects-00/risc/pipelining/index.html (2007 05 24)

[21] http://www.es.ele.tue.nl/mininoc/doc/jpeg_mmips.htm (2007 05 20)

[22] http://www.es.ele.tue.nl/mininoc/doc/module_network2x2.htm (2007 05 20)

[23] http://www.sei.cmu.edu/productlines/about_logo.html (2007 05 25)