

Laboratorinis darbas. Registrai

Turinys

1 Tikslas	2
2 Teorija	2
3 Laboratorinio darbo užduotis	7
4 Pavyzdys	8
4.1 PLIS matricos programavimo pavyzdys	11

1. Tikslas

Susipažinti su įvairių tipų registrais, jų struktūra, veikimu, taikymo galimybėmis ir realizavimu naudojant trigerius. Išsiaiškinti postūmių operacijas ir jų atlikimo būdus. Patikrinti jų veikimą programuojamos logikos schemeje.

2. Teorija

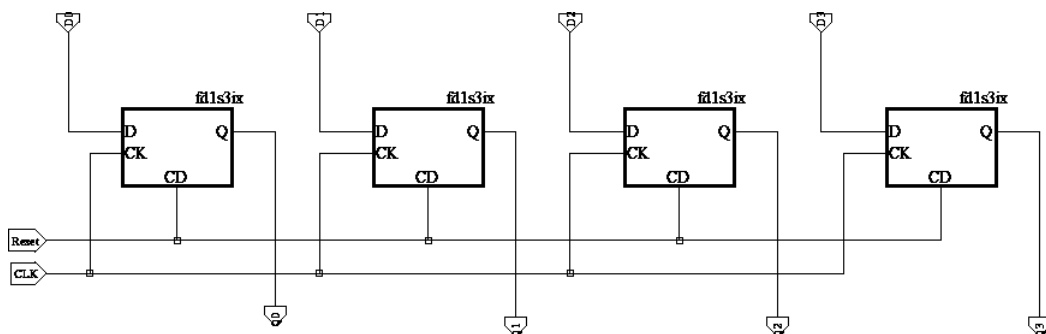
Registras – tai įtaisas, skirtas informacijai įrašyti, saugoti ir skaityti, taip pat kitoms operacijoms su informacija atlikti. Registras gali saugoti daugiau nei vieną bitą informacijos. Registrus sudaro atminties ląstelės – trigeriai ir juos valdančios schemas. Pagal informacijos įvedimo ir išvedimo būdą registras gali būti:

- saugojimo (lygiagretusis);
- postūmio (nuoseklusis);
- universalusis.

Registras gali būti vientaktis (dažniausiai naudojami dinaminio valdymo trigeriai) ar dvitaktis, kai informacija įrašoma per du taktus (kaip ir MS trigeriuose). Kadangi žemiausiąją informacijos žodžio ar baido skiltį (nulinį indeksą) įprasta žymėti dešinėje, mes registruose saugomos informacijos skiltis žymėti pradėsime nuo dešinėsios pusės (literatūroje kartais žymima ir atvirkščiai). Tokiu atveju registre įrašytą informaciją verčiant į dešimtainę sistemą galima dvejetą kelti indekso laipsniu.

2.0.1 Lygiagretieji registrai

Lygiagretųjų registrą galima sudaryti iš n sinchroninių D trigerių su įvesties ir išvesties valdymo grandinėmis, kaip parodyta 1 pav.



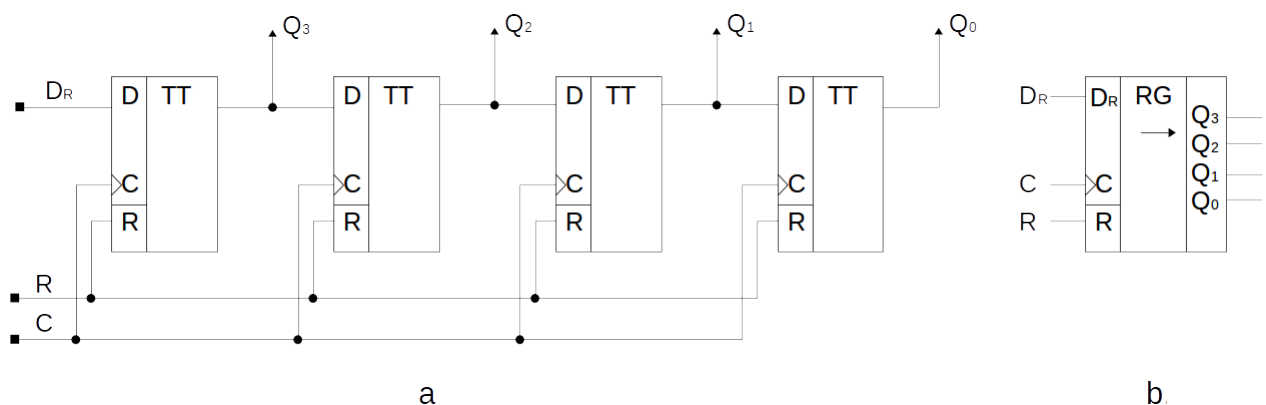
1 pav. Keturių skilčių lygiagrečiojo registro su valdymo grandinėmis schema

2.1 Postūmio (nuoseklus) registras

D trigeriuose esantis įvesties signalas R ištrina įrašytą informaciją ir visus trigerius nustato į nulinę būseną. Signalas ID valdo įrašomą informaciją. Jei $ID = 1$, informacija bus įrašoma tiesioginiu kodu. Jei $ID = 0$, informacija bus įrašoma atvirkštiniu kodu (invertuota). Signalas RD valdo įrašytos informacijos skaitymą. Jei $RD = 1$, registre saugoma informacija į išvestį siunčiama tiesioginiu kodu. Jei $RD = 0$, registre saugoma informacija į išvestį siunčiama atvirkštiniu kodu.

2.1 Postūmio (nuoseklus) registras

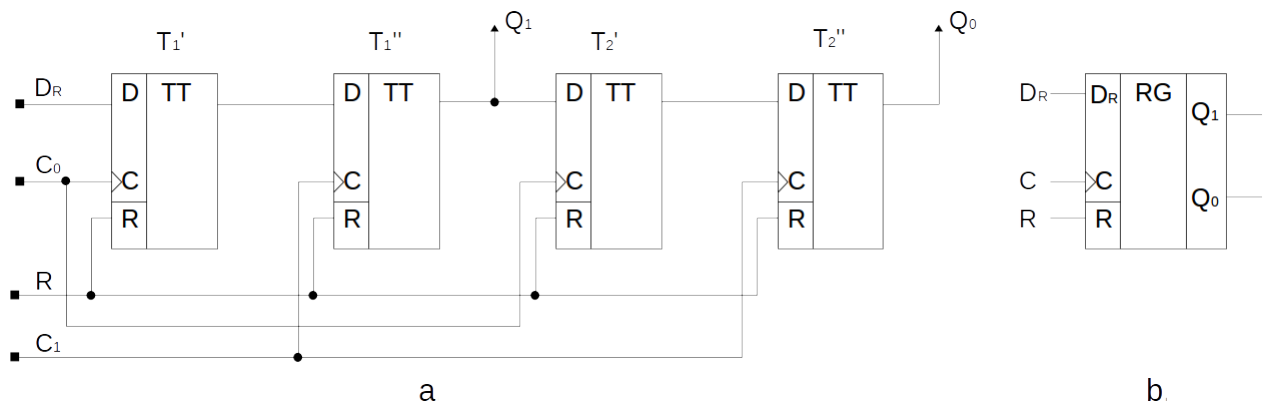
Postūmio registras vykdo postūmį į dešinę (LR , tiesioginis), į kairę (LL , reversinis), į dešinę arba į kairę (universalus). Vykdamas postūmį, kiekvienu sinchroimpulso taktu įveda ma informacija nuosekliai stumia išilgai registro, kol pasiekia registro išėjimą. Nuosekliai sujungę sinchroninius dinaminio valdymo trigerius, gausime postūmio registrą, vykdamą loginį postūmį į dešinę. Tokio registro schema pateikta 2 pav.



2 pav. Keturių skilčių postūmio į dešinę registro schema (a) ir grafinis žymėjimas (b)

Panaudoję statinio valdymo D trigerius, gausime dvitaktį postūmio registrą, kurio schema pateikta 3 pav. Pirmojo takto metu informacija įrašoma į pirmuosius trigerius T'_1, T'_2 , antro takto metu – į antruosius trigerius T''_1, T''_2 . Dažnai į C_1 siunčiamas tiesioginis sinchronizavimo impulsas C , o į C_2 – inversinis (\overline{C}).

2.2 Neigiamųjų skaičių saugojimas registruose



3 pav. Dviejų skilčių dvitakčio postūmio į dešinę registro schema (a) ir jos grafinis žymėjimas (b)

Skiriami trys postūmių tipai:

1. Loginis postūmis (LL , LR), atliekamas į laisvąsias skiltis įrašant signalą atėjusį į specialią įvestį (D_L , D_R);
2. Ciklinis postūmis (CR , CL), atliekamas į laisvąsias skiltis įrašant informaciją iš išstumtųjų skilčių;
3. Aritmetinis postūmis (AR , AL), atliekamas aukščiausiosios (ženklų) skilties nestumiant ir tokiu būdu išsaugant skaičiaus ženklą. Likusių skilčių postūmis viena skiltimi į dešinę (AR_1) reiškia skaičiaus absoliučiosios reikšmės sumažinimą du kartus, į kairę – padidinimą du kartus. Atlikdami aritmetinį postūmį, į laisvąsias skiltis įrašome 0 arba 1, priklausomai nuo skaičiaus kodo ir ženklo.

2.2 Neigiamųjų skaičių saugojimas registruose

Skaitmeninėje logikoje kodavimas reikalingas norint dvejetainė sistema atvaizduoti neigiamuosius skaičius. Neigiamieji skaičiai bet kokioje skaičiavimo sistemoje vaizduojami pridedant prie jų minuso ženklą. Skaitmeninėse schemose skaičiai saugomi bitais ir vietos papildomiems simboliams nėra. Dažniausiai taikomi trys saugojimo metodai:

1. Tiesioginis kodas (angl. *Sign-and-magnitude*);
2. Atvirkštinis kodas (angl. *Ones' complement*);
3. Papildomas kodas (angl. *Two's complement*).

Yra ir daugiau būdų neigiamiems skaičiams saugoti, tačiau jie taikomi retai. Dažniausiai ir beveik visuose šiuo metu naudojamuose bendrosios paskirties procesoriuose vartojamas papildomas kodas (tačiau nėra kriterijaus, pagal kurį kuris nors vienas kodas būtų geresnis už kitus).

Tiesioginis kodas. Šiuo atveju ženklui saugoti skiriamas vienas registro bitas. Dažniausiai tai aukščiausias bitas. Jei šis bitas: 0 – skaičius teigiamas, 1 – skaičius neigiamas; Likę registro bitai aprašo skaičiaus absoliučiąją vertę (dydį). Taigi, jei registre 8 bitai, vienas jų skiriamas ženklui. Saugoti vertei lieka 7. Skaičiaus vertė gali kisti nuo 0000000 (0) iki 1111111 (127). Vadinasi, tokiam registre galima atvaizduoti skaičius nuo -127 iki $+127$. Šioje sistemoje yra du būdai aprašyti skaičių 0: 00000000 (+0) ir 10000000 (–0). Šis metodas – tiesioginis būdas parodyti ženklą (pridėti + arba – prie skaičiaus). Kai kurie ankstyvieji kompiuteriai (pvz., IBM 7090) taikė šį neigiamųjų skaičių saugojimo metodą dėl jo intuityvumo.

Pvz., Tiesioginiame kode skaičius:

+26 : 00011010

–26 : 10011010

Atvirkštinis kodas. Šis būdas vadinamas atvirkštiniu kodu, nes neigiamieji skaičiai jame atvaizduojami kaip teigiamųjų skaičių bitų inversija. Kaip ir tiesioginis kodas, atvirkštinis turi du būdus atvaizduoti skaičių 0: 00000000 (+0) ir 11111111 (–0). Pvz., Atvirkštiniame kode skaičius:

+26 : 00011010

–26 : 11100101

Į N bitų ilgio registrą galima išsaugoti skaičius nuo $-(2^{N-1} - 1)$ iki $+(2^{N-1} - 1)$ ir ± 0 .

Papildomas kodas. Norint išspręsti keleto skaičiaus 0 žymėjimo ir pernašos pridėjimo problemas, sugalvota sistema, vadinama papildomu kodu. Papildomame kode neigiamieji skaičiai atvaizduojami bitų seka, didesne vienetu nei atvirkštinis teigiamojo skaičiaus kodas. Papildomame kode yra tik vienas būdas atvaizduoti nulį (00000000). Priešingas skaičius (nesvarbu, teigiamas ar neigiamas) randamas invertuojant visus bitus ir prie rezultato pridant vienetą. Šiame kode neigiamieji ir teigiamieji skaičiai sudedami vienodai, nėra reikalo papildomai pridėti pernašų. Atimtis taip pat atliekama vienodai. Neigiamasis skaičius verčiamas priešingu (teigiamuoju) taip:

1. Invertuojami visi šio skaičiaus bitai;

2.3 Universalusis registras

2. Pridedamas vienetas;

Pvz., +1, dvejetainėje sistemoje 000000001 :

1. $\overline{00000001} \rightarrow 11111110$

2. $11111110 + 1 \rightarrow 11111111$ (-1 papildomame kode)

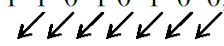
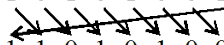
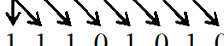

Tokia sistema naudinga supaprastinant skaičiavimus aparatuose: ji leidžia sudėti neigiamuosius skaičius nenaudojant atimties operacijos ir neatliekant jokių papildomų korekcijos veiksmų. Nereikia ir logikos, nustatančios skaičiaus ženklą. Ta pačia logika, kuria atliekama sudėtis, galima atlikti ir atimtį.

Atliekant informacijos postūmius registruose, į laisvas skiltis įrašoma informacija, pa-
vaizduota 1 lentelėje.

1 lentelė. Laisvų skilčių užpildymas atliekant poslinkius

Ženklas (skaičius)	Kodas	Į laisvą skiltį įrašoma	
		AR stumiant į dešinę	AL stumiant į kairę
+ teigiamas	bet koks	0	0
	tiesioginis	0	0
- neigiamas	atvirkštinis	1	1
	papildomas	1	0

Pavyzdys. Turime

- a) Atlikus postūmį $LL_1, 1$ $R = 1 \ 1 \ 0 \ 1 \ 0 \ 1 \ 0 \ 0;$

 $R = 1 \ 0 \ 1 \ 0 \ 1 \ 0 \ 0 \ 1 \leftarrow DL;$

b) Atlikus postūmį CR_1 $R = 1 \ 1 \ 0 \ 1 \ 0 \ 1 \ 0 \ 0;$

c) Atlikus postūmį AR_1 atv. kode $R = 1 \ 1 \ 1 \ 0 \ 1 \ 0 \ 1 \ 0.$


2.3 Universalusis registras

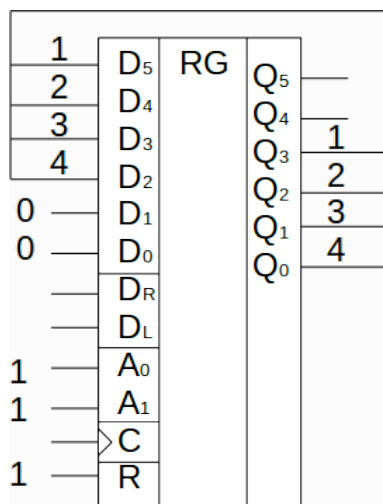
Postūmio registras paprastai vykdo loginį postūmį per vieną skiltį į dešinę (LR_1) ar į kairę (LL_1) ir lygiagretų informacijos įrašymą. Informacijai įrašyti į atsilaisvinusią skiltį naudojama atskira įvestis D_R (vykdant LR_1) ir D_L (vykdant LL_1). Toks registras vadinamas universaliuoju. Šie registrai gali realizuoti ir ciklinį postūmį į dešinę ar į kairę. Tada pakanka išstumiamosios skilties išvestį sujungti su atitinkama D_R ar D_L įvestimi. Kitas postūmio operacijas (pavyzdžiui, postūmį per dvi skiltis) galima realizuoti naudojant lygiagretų įrašymą, tada registro išvestys sujungiamos su atitinkamomis įvestimis.

Pavyzdžiui, turime 6 skilčių universalųjį postūmio registrą, vykdančią loginio postūmio per 1 skiltį į dešinę, į kairę ir lygiagretų informacijos įrašymą. Šio registro teisingumo lentelė pateikta 2 lentelėje.

2 lentelė. Universaliojo registro veikimo lentelė

R	A_0A_1	D_RD_L	$D_5...D_0$	Q_5	Q_4	Q_3	Q_2	Q_1	Q_0	Paaškinimai
0	$x\ x$	$x\ x$	$x...x$	0	0	0	0	0	0	Nulio nustatymas
1	0 0	$x\ x$	$x...x$	Q_5	Q_4	Q_3	Q_2	Q_1	Q_0	Saugojimas
1	1 0	1 x	$x...x$	1	Q_5	Q_4	Q_3	Q_2	Q_1	Loginis postūmis į dešinę, įrašant $D_R(LR_1, D_R)$
1	1 0	0 x	$x...x$	0	Q_5	Q_4	Q_3	Q_2	Q_1	
1	0 1	$x\ 1$	$x...x$	Q_4	Q_3	Q_2	Q_1	Q_0	1	Loginis postūmis į kairę, įrašant $D_L(LL_1, D_L)$
1	0 1	$x\ 0$	$x...x$	Q_4	Q_3	Q_2	Q_1	Q_0	0	
1	1 1	$x\ x$	$D_5...D_0$	D_5	D_4	D_3	D_2	D_1	D_0	Lygiagretus informacijos įrašymas

Universaliojo registru galima realizuoti bet kokią informacijos postūmį. Pavyzdžiui, loginį postūmį per dvi skiltis įrašant nulius ($LL_2, 0$). Šį postūmį realizuos schema, pavaizduota 4 pav.



4 pav. Loginis postūmis universaliojo registru į kairę per dvi skiltis įrašant 0 ($LL_2, 0$)

3. Laboratorinio darbo užduotis

1. Naudojant scheminį redaktorių sudaryti užduotyje, (žr. individualių užduočių lentelę) nurodyto ilgio postūmio registro schemą (žr. 2 lentelę). Sudaryti testinius rinkinius ir patikrinti, kaip veikia schema.

-
2. Naudojant multiplexerius ir lygiagretųjį registrą suprojektuoti specializuotą postūmio registrą, realizuojantį užduotyje nurodytas mikrooperacijas. Sudaryti testinius rinkinius ir patikrinti, kaip veikia schema.
 3. Naudojant sukurta specializuoto registro schemą suprogramuoti loginę PLIS matricą. Patikrinti, kaip matrica veikia laboratoriniame stende.
 4. Parengti laboratorinio darbo ataskaitą. Joje pateikti suprojektuotas schemas ir laiko diagramas. Taip pat pateikti PLIS matricai programuoti parengtą schemą. Aprašyti, kaip schema veikia stende.

4. Pavyzdys

Skilčių skaičius: 5; postūmio mikrooperacijos: LL_2 , CR_1 , AL_2 ; įrašoma informacija: 1; nulinio nustatymas: sinchroninis; skilčių kodas: papildomas.

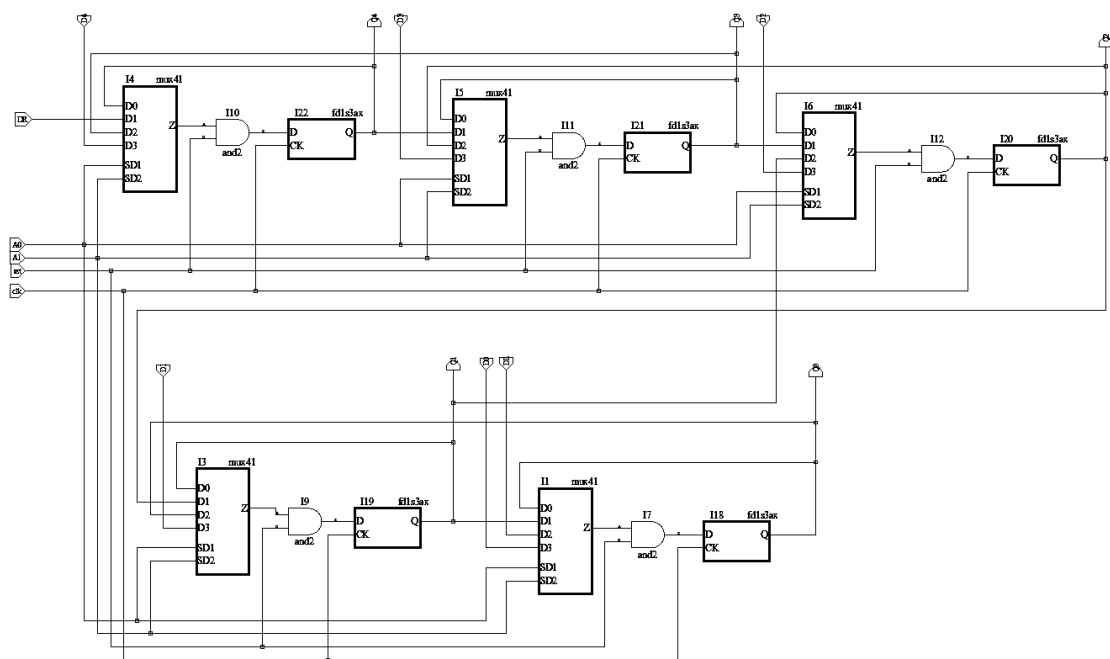
1. Sudarome universaliojo registro veikimo lentelę:

R	A_0A_1	Q_4	Q_3	Q_2	Q_1	Q_0	Paaškinimai
0	$x \ x$	0	0	0	0	0	Nulinio nustatymas
1	0 0	Q_4	Q_3	Q_2	Q_1	Q_0	Saugojimas
1	1 0	D_R	Q_4	Q_3	Q_2	Q_1	Loginis postūmis į dešinę, įrašant $D_R(LR_1, D_R)$
1	0 1	Q_3	Q_2	Q_1	Q_0	D_L	Loginis postūmis į kairę, įrašant $D_L(LL_1, D_L)$
1	1 1	D_4	D_3	D_2	D_1	D_0	Lygiagretus informacijos įrašymas

Registras vykdo keturias mikrooperacijas, todėl joms valdyti naudojame multiplexerius. Signalus A_1 , A_0 sujungiame į multiplexerių adresines įvestis, o duomenų įvestis sujungiame su signalais naudodamiesi registro veikimo lentele.

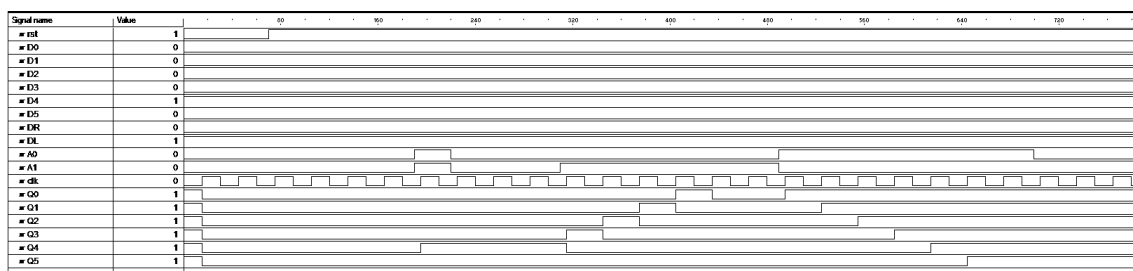
- **Sinchroniniam** nulinio nustatymui naudosime bibliotekoje esančius sinchroninius D trigerius (D Flip-Flop GSR Clear) **fd1s3ax**
- **Asinchroniniam** nulinio nustatymui naudosime D trigerius (D Flip-Flop Asynchronous clear) su asinchrone **RESET** įvestimi **fd1s3dx**

Sinchroniniam nulinio nustatymui prieš trigerių D įvestis įterpkime IR elementus, valdomus signalo **RESET** (asinchroniniam nulinio nustatymui parenkame atitinkamus trigerius). Universaliojo registro schema parodyta 5 pav.



5 pav. Universaliojo registro schema

Sudarome registro veikimą tikrinantį testą, kuris patikrintų visas registro vykdomas mikrooperacijas. Suvedę *.vhd failą ir sukompiliavę testinius vektorius gauname laikines diagramas, pavaizduotas 6 pav.



6 pav. Universaliojo registro laikinė diagrama

2. Projektuojant specializuotą registrą, vykdančią postūmio mikrooperacijos LL_2 įrašant 1, CR_1 ir AL_2 naudosime multiplexerius su keturiomis įvestimis ir trigerius. Multiplexerius valdo du įvesties signalai: A_0 ir A_1 , atitinkantys mikrooperacijų kodus. Schemai sudaryti reikalingą informaciją surašome į 3 lentelę.

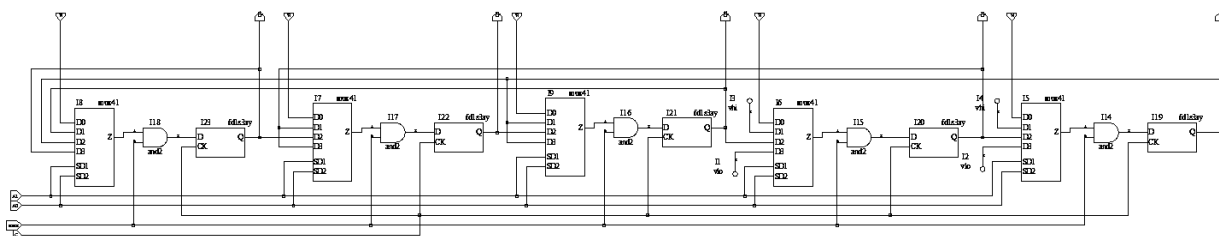
3 lentelė. Specializuoto registro mikrooperacijos

MO kodas							MO paaiškinimas
A_0	A_1	D_4	D_3	D_2	D_1	D_0	
0	0	x_4	x_3	x_2	x_1	x_0	Informacijos įrašymas
0	1	Q_2	Q_1	Q_0	1	1	$LL_2, 1$
1	0	Q_0	Q_4	Q_3	Q_2	Q_1	CR_1
1	1	Q_4	Q_1	Q_0	0	0	AL_2 , pap. kodas

Lentelę atitinka Bulio funkcijų sistema:

$$\begin{cases} D_0 = \overline{A_0}\overline{A_1}x_0 \cup \overline{A_0}A_11 \cup A_0\overline{A_1}Q_1 \cup A_0A_10; \\ D_1 = \overline{A_0}\overline{A_1}x_1 \cup \overline{A_0}A_11 \cup A_0\overline{A_1}Q_2 \cup A_0A_10; \\ D_2 = \overline{A_0}\overline{A_1}x_2 \cup \overline{A_0}A_1Q_0 \cup A_0\overline{A_1}Q_3 \cup A_0A_1Q_0; \\ D_3 = \overline{A_0}\overline{A_1}x_3 \cup \overline{A_0}A_1Q_1 \cup A_0\overline{A_1}Q_4 \cup A_0A_1Q_1; \\ D_4 = \overline{A_0}\overline{A_1}x_4 \cup \overline{A_0}A_1Q_2 \cup A_0\overline{A_1}Q_0 \cup A_0A_1Q_4; \end{cases}$$

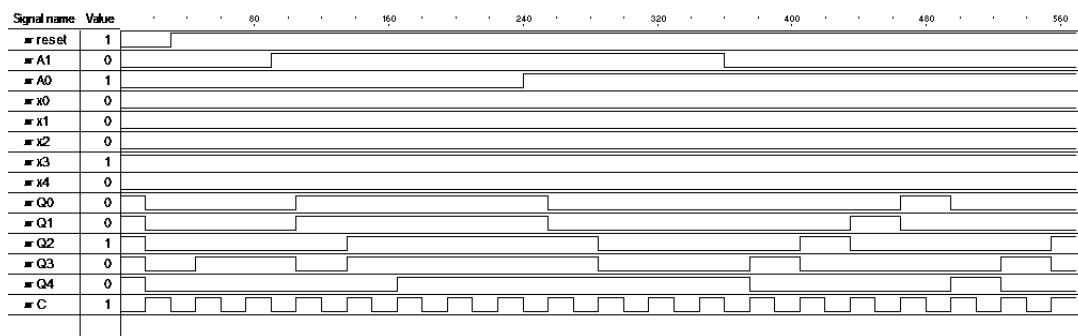
Šią sistemą patogiu realizuoti multiplekseriais, kaip pavaizduota 7 pav.



7 pav. Sinchroninio nulinio nustatymo specializuoto registro schema

3. Sudarome registro veikimą tikrinantį testą, kuris patikrintų visas registro vykdomas mikrooperacijas. Suvedę *.vhd failą ir sukompiliavę testinius vektorius, gauname 8 pav. parodytas laikines diagramas.

4.1 PLIS matricos programavimo pavyzdys



8 pav. Specializuoto registro laikinė diagrama

Nagrinėdami laikines diagramas matome, kad specializuotas registras vykdo informacijos įrašymo ir LL_2 , CR_1 , AL_2 mikrooperacijas.

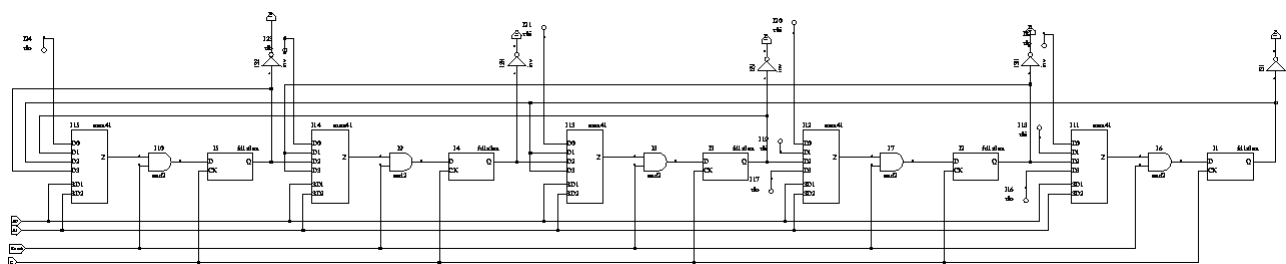
4.1 PLIS matricos programavimo pavyzdys

4.1.1 Lattice Brevia laboratorinis stendas

Programuoti Lattice Brevia naudosime laboratorinio darbo metu suprojektuotą specializuoto registro schemą. (7 pav.).

PLIS matricai programuoti schemą pakeisime:

- Panaikiname $x1 - x6$ įvestis nustatę atitinkamą įkėlimo kombinaciją.
- Invertuojame visus išvesčių signalus ($Q0 - Q4$), kadangi LED diodai šviečia padavus žemą loginį lygį. Pakeista schema pavaizduota 9 pav.



9 pav. Specializuoto registro schema, pritaikyta PLIS matricai

Fiziniam kontaktams priskirti naudojame *Spreadsheet view* įrankį (**Tools**→**Spreadsheet View**). Šio įrankio langas pavaizduotas 10 pav. Pasinaudoję žymėjimais ant plokštės, priskiriame įvestis ir išvestis. Mygtukų bei LED diodų kontaktams būtina nustatyti **PULLMODE** parametą **UP**. Nepaspausti šie mygtukai išduos aukštą loginį lygį.

4.1 PLIS matricos programavimo pavyzdys

Spreadsheet View												
	Name	Group By	Pin	BANK	VREF	IO_TYPE	PULLMODE	DRIVE	SLEWRATE	PCICLAMP	OPENDRAIN	Outload (pF)
1	All Ports	N/A	N/A	N/A	N/A		N/A	N/A	N/A	N/A	N/A	N/A
1.1	Input	N/A	N/A	N/A	N/A		N/A	N/A	N/A	N/A	N/A	N/A
1.1.1	Reset	N/A	50(50)	5(5)	N/A	LVC MOS2...	UP(UP)	NA(NA)	FAST(FAST)	OFF(OFF)	OFF(OFF)	N/A
1.1.2	Clock	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
1.1.2.1	C	N/A	52(52)	5(5)	N/A	LVC MOS2...	UP(UP)	NA(NA)	FAST(FAST)	OFF(OFF)	OFF(OFF)	N/A
1.1.3	A1	N/A	55(55)	5(5)	N/A	LVC MOS2...	UP(UP)	NA(NA)	FAST(FAST)	OFF(OFF)	OFF(OFF)	N/A
1.1.4	A0	N/A	56(56)	5(5)	N/A	LVC MOS2...	UP(UP)	NA(NA)	FAST(FAST)	OFF(OFF)	OFF(OFF)	N/A
1.2	Output	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
1.2.1	D4	N/A	43(43)	5(5)	N/A	LVC MOS2...	UP(UP)	12(12)	FAST(FAST)	OFF(OFF)	OFF(OFF)	0.000
1.2.2	D3	N/A	40(40)	5(5)	N/A	LVC MOS2...	UP(UP)	12(12)	FAST(FAST)	OFF(OFF)	OFF(OFF)	0.000
1.2.3	D2	N/A	39(39)	5(5)	N/A	LVC MOS2...	UP(UP)	12(12)	FAST(FAST)	OFF(OFF)	OFF(OFF)	0.000
1.2.4	D1	N/A	38(38)	5(5)	N/A	LVC MOS2...	UP(UP)	12(12)	FAST(FAST)	OFF(OFF)	OFF(OFF)	0.000
1.2.5	D0	N/A	37(37)	5(5)	N/A	LVC MOS2...	UP(UP)	12(12)	FAST(FAST)	OFF(OFF)	OFF(OFF)	0.000

10 pav. *Spreadsheet View* langas su priskirtais prievadais

Išsaugojus pakeitimus **Spreadsheet View** lange, projekte automatiškai sukuriamas apribojimų (constraints) *.lpf failas. Šio failo turinys parodytas kodo pavyzdyje. Atliekant pakeitimus scheme šis failas automatiškai neatnaujinamas, todėl jei pakeičiamas įvesčių ar išvesčių skaičius ar pavadinimai, būtina nereikalingas eilutes iš šio failo ištrinti ranka (prievadų priskyrimą reikės atlikti tik naujoms ar pervadintoms įvestims/išvestims) arba ištrinti visą failą (prievadų priskyrimą *Spreadsheet view* tokiu atveju reikės atlikti iš naujo).

```

BLOCK RESETPATHS ;
BLOCK ASYNCPATHS ;
LOCATE COMP "C" SITE "52" ;
LOCATE COMP "A1" SITE "55" ;
LOCATE COMP "A0" SITE "56" ;
LOCATE COMP "D0" SITE "37" ;
LOCATE COMP "D1" SITE "38" ;
LOCATE COMP "D2" SITE "39" ;
LOCATE COMP "D3" SITE "40" ;
LOCATE COMP "D4" SITE "43" ;
LOCATE COMP "Reset" SITE "50" ;

```

Schemas fizinis išdėstymas (planavimas) plokštėje atliekamas **Place and route** meniu punktu. Atliekame **Place and Route** žingsnį iš **Process** meniu. Paskui generuojame PLIS programavimo failą. Varnele pažymime **Process** lango **Export Files** skiltyje esantį **JEDEC File** punktą.

Pasirinkus **JEDEC** įvykdomas **Export Files** punktas. Projekte atsiras *projekto_vardas.jed* failas. Šis failas ir naudojamas matricai programuoti. Matrica prijungiama prie USB prievado ir pagrindiniame meniu pasirenkama **Tools→ Programmer**. Atsidariusiame programatoriaus lange patikriname, ar **Device Family** bei **Device** atitinka turimą matricą (Lattice Brevia laboratorinio stendo atveju tai *LatticeXP2-5E-6TN144C*), jei neatitinka – projekto kūrimo metu pasirinkta ne ta biblioteka. Taip pat įsitikiname, kad *File Name* skiltyje nurodytas reikiamas (šio projekto) *.jed failas.

Matricai programuoti naudojame pagrindinio meniu komandą **Design→Program** ar-

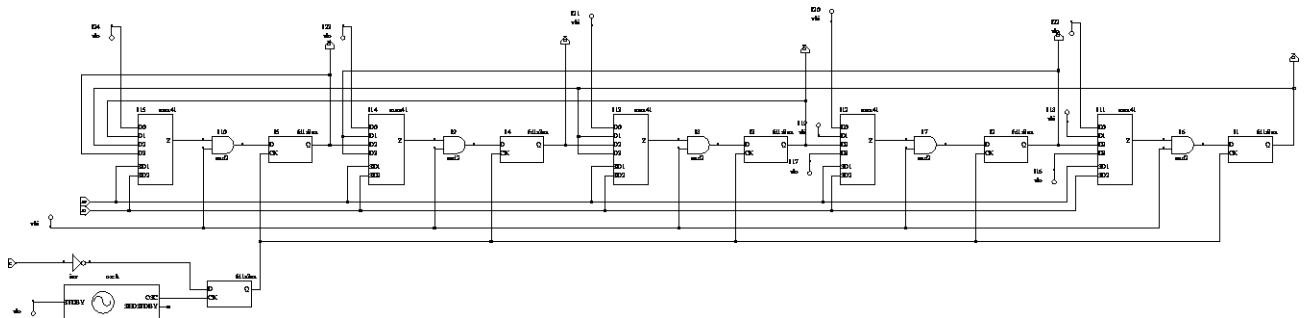
ba **Program** mygtuką programatoriaus lange. Toliau patikriname, ar schemos veikimas atitinka modeliavimo rezultatus.

4.1.2 Lattice MachXO2 laboratorinis stendas

Programuoti PLIS naudosime laboratorinio darbo metu suprojektuotą specializuoto registro schemą.

PLIS matricai programuoti schemą pakeisime:

- Norėdami prie mygtuko prijungti sinchronizavimo signalą be trikdžių, schemą papildome PLIS vidiniu osciliatoriumi (komponentas *OSCH*), D trigeriu ir inverteriu.
- Panaikiname *RESET* įvestį. Nustatome jame žemą signalo lygį, panaudodami komponentą *VLO*.
- Panaikiname $x0 - x4$ įvestis nustatę atitinkamą įkėlimo kombinaciją. Pakeista schema pavaizduota 11 pav.



11 pav. Specializuoto registro schema, pritaikyta PLIS matricai su pažymėtais pakeitimais

Fiziniais kontaktams priskirti naudojame *Spreadsheet view* įrankį (**Tools**→**Spreadsheet View**). Šio įrankio langas pavaizduotas 12 pav. Pasinaudoję grafiniais žymėjimais ant plokštės priskiriame įvestis ir išvestis. B1 ir C3 kontaktams, einantiems į mygtukus, būtina nustatyti **PULLMODE** parametą **UP**. Nepaspausti šie mygtukai išduos aukštą loginį lygį.

4.1 PLIS matricos programavimo pavyzdys

	Name	Group By	Pin	BANK	BANK_VCC	VREF	IO_TYPE	PULLMODE	I
1	All Ports	N/A	N/A	N/A	N/A	N/A			N/A
1.1	Input	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
1.1.1	A0	N/A	B1(B1)	3(3)	Auto	N/A	LVC MOS2...	DOWN(DOWN)	NA
1.1.2	A1	N/A	C3(C3)	3(3)	Auto	N/A	LVC MOS2...	DOWN(DOWN)	NA
1.1.3	C	N/A	N3(N3)	2(2)	Auto	N/A	LVC MOS2...	DOWN(DOWN)	NA
1.2	Output	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
1.2.1	D0	N/A	D1(D1)	3(3)	Auto	N/A	LVC MOS2...	DOWN(DOWN)	8(8)
1.2.2	D1	N/A	B8(B8)	0(0)	Auto	N/A	LVC MOS2...	DOWN(DOWN)	8(8)
1.2.3	D2	N/A	C8(C8)	0(0)	Auto	N/A	LVC MOS2...	DOWN(DOWN)	8(8)
1.2.4	D3	N/A	B13(...)	0(0)	Auto	N/A	LVC MOS2...	DOWN(DOWN)	8(8)
1.2.5	D4	N/A	A13(...)	0(0)	Auto	N/A	LVC MOS2...	DOWN(DOWN)	8(8)

12 pav. *Spreadsheet View* langas su priskirtais prievadais

Išsaugojus pakeitimus **Spreadsheet View** lange, projekte automatiškai sukuriamas apribojimų (constraints) *.lpf failas. Šio failo turinys parodytas toliau. Atliekant pakeitimus schemoje šis failas automatiškai neatnaujinamas, todėl jei pakeičiamas įvesčių ar išvesčių skaičius ar pavadinimai, būtina nereikalingas eilutes iš šio failo ištrinti ranka (prievadų priskyrimą reikės atlikti tik naujoms ar pervadintoms įvestims/išvestims) arba ištrinti visą failą (prievadų priskyrimą *Spreadsheet view* tokiu atveju reikės atlikti iš naujo).

```

BLOCK RESETPATHS ;
BLOCK ASYNCPATHS ;
LOCATE COMP "A0" SITE "B1" ;
LOCATE COMP "A1" SITE "C3" ;
LOCATE COMP "C" SITE "N3" ;
LOCATE COMP "D0" SITE "D1" ;
LOCATE COMP "D1" SITE "B8" ;
LOCATE COMP "D2" SITE "C8" ;
LOCATE COMP "D3" SITE "B13" ;
LOCATE COMP "D4" SITE "A13" ;

```

Schemas fizinis išdėstymas (planavimas) plokštėje atliekamas **Place and route** meniu punktu. Atliekame **Place and Route** žingsnį iš **Process** meniu. Paskui generuojame PLIS programavimo failą. Varnele pažymime **Process** lango **Export Files** skiltyje esantį **JEDEC File** punktą.

Pasirinkus **JEDEC** įvykdomas **Export Files** punktas. Projekte atsiras *projekto_vardas.jed* failas. Šis failas ir naudojamas matricai programuoti. Matrica prijungiama prie USB prievado ir pagrindiniame meniu pasirenkama **Tools**→ **Programmer**. Atsidariusiame programatoriaus lange patikriname, ar **Device Family** bei **Device** atitinka turimą matricą (laboratorinio darbo metu tai *MachXO2-LCMXO2-1200ZE*), jei neatitinka – projekto

4.1 PLIS matricos programavimo pavyzdys

kūrimo metu pasirinkta ne ta biblioteka. Taip pat įsitikiname, kad *File Name* skiltyje nurodytas reikiamas (šio projekto) *.jed failas.

Matricai programuoti naudojame pagrindinio meniu komandą **Design→Program** arba **Program** mygtuką programatoriaus lange. Toliau patikriname, ar schemos veikimas atitinka modeliavimo rezultatus.