Kauno technologijos universitetas

Informatikos Fakultetas

4 Laboratorinio Darbo Ataskaita

P175B100 Skaitmeninės Logikos Pradmenys

Atliko:

IFF-1/9 grupės studentas
Nedas Liaudanskis

Priėmė:

Lekt. Jurgita Arnastauskaitė

Kaunas, 2022

1. Įvadas

1.1 Tikslas

Perprasti įvairiausių daliklių ir skaitiklių veikimo principus, VHDL kalbos kode, projektuojant ir taikant įvairius metodus ir kodo eilutes. Įrodyti jų veikimą simuliacijoje ir programuojamos logikos schemoje.

1.2 Užduotis

Užduoties numeris: 130

Užduoties registrų sąlygos:

130	20	41	150	389	1656

Teorija:

- Skaitiklis, tai operacinis komponentas, kuris prideda arba atima skaitiklį iš turimo skaičiaus. Skaitikliai, kuriuos mes kursime yra sudaryti iš trigerių. Šie skaitikliai turi skaitiklio modulį, kuris yra lygus trigerių skaičiui. Modulis rodo iki kiek gali skaičiuoti mūsų skaitiklis.
- Pasiekus skaitiklio modulį įvyksta pernaša, kuri parodo, jog skaitiklis pasiekė skaičiavimo ribą. Šį signalą galime naudoti kitų signalų įvestyse. Visi skaitikliai taip pat turi savo skirstymus: asinchroninius ir sinchroninius.
- Asinchroninių skaitiklių veikimo principas: Visus ateinančius signalus siunčia praeitas trigeris, dėl šios priežasties visi skaitiklio trigeriai persijungia nuosekliai. Šių trigerių struktūra sudaryta iš D trigerių.
- Sinchroninių skaitiklių veikimo principas: Visi trigeriai persijungia vienu metu, dėl to jog yra naudojamas tik vienas signalas, visiems esamiems trigeriams.

2. Formulės ir skaičiavimai

Pirmiausia visus turimus skaičius pasiverčiau į dvejetainius, taip sužinojau kiek trigerių reiks norint saugoti informacija kiekvienoje schemoje.

M1	M2	M3
20	41	150
10100	101001	10010110

Tada pasinaudodamas formules apsiskaičiavau reikiamus dydžius, kurie bus naudojami schemose, kai reikės žinoti būsenas, kurios metu JM skaitiklyje įvyks pernašos:

$$S2 = (JM - 1) div\ M1 = \frac{388}{20} = 19 \text{ (antrojo skaitiklio būsena)}$$

$$S1 = (JM - 1) mod\ M1 = 388 mod\ 20 = 8 \text{ (pirmojo skaitiklio būsena)}$$

Apskaičiavęs JM skaitiklio būsenas, naudodamas kitas formules apskaičiavau JM2 skaitiklio būsenas, kurių metu įvyksta pernašos.

$$S3 = (JM2 - 1) div (M1 * M2) = 1655 div (20 * 41) = 2$$

$$O1 = (JM2 - 1) mod (M1 * M2) = 1655 mod (20 * 41) = 15$$

$$S2 = O1\ div M1 = 15 div 20 = 0$$

$$S1 = O1\ mod M1 = 15 mod 20 = 15$$

S1, S2, S3 – skaitiklių būsenos atitinkamu metu.

3. M1 skaitiklio realizacija

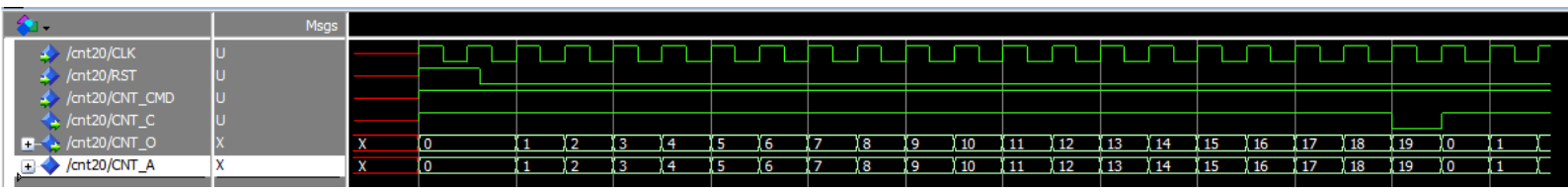
3.1 M1 skaitiklio vhdl kodas

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity CNT20 is port (
        CLK         : in std_logic; --Sinchro signalas
        RST         : in std_logic; -- Reset signalas
        CNT_CMD     : in std_logic; -- Komanda
        CNT_C       : out std_logic; --Pernasa
        CNT_O       : out std_logic_vector(4 downto  0) --trigerių skaičius
        );
end CNT20;


architecture rtl of CNT20 is
    signal CNT_A: unsigned (4 downto  0);
begin
    process(CLK, RST, CNT_CMD)
    begin
        if RST = '1' then -- kai RST, reset signalas 1, visi trigeriai nustatomi į 0
            CNT_A <= "00000";
            CNT_C <= '1';
        elsif CLK'event and CLK = '1' and CNT_CMD = '1' then --jeigu reset nėra 1 tada yra patikrinama ar dabartinė padėtis yra mažesnė už pernašą
            if CNT_A < 19  then
                CNT_A <= CNT_A + 1; --Prideda vienetą prie skaitiklio
                if CNT_A = 18 then
                    CNT_C <= '0';
                else
                    CNT_C <= '1';
                end if;
            else
                CNT_C <= '1'; -- jeigu dabartinė padėtis yra pernaša, kitas veiksmas nustato visus trigerius į
                CNT_A <= "00000";
            end if;
        end if;
    end process;
CNT_O <= std_logic_vector(CNT_A);
```

3.2 Testavimo direktyvos

```
force -freeze sim:/cnt20/CLK 1 0, 0 {50 ps} -r 100

force -freeze sim:/cnt20/RST 1 0

force -freeze sim:/cnt20/CNT_CMD 1 0

run

run

force -freeze sim:/cnt20/RST 0 0

run
```

3.3 M1 skaitiklio simuliacija

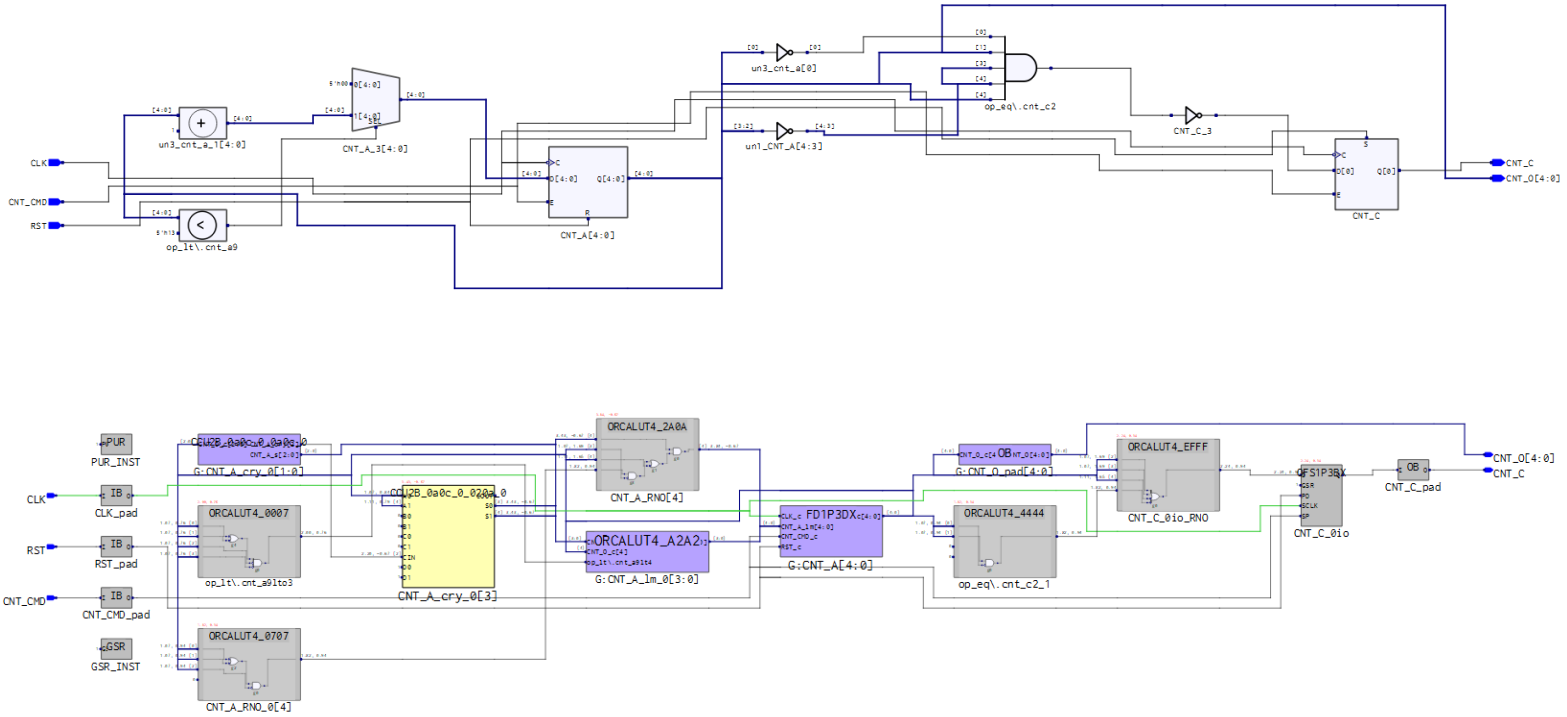


Aptarimas:

- Pirmiausia, nustatome clock(CLK) ir CNT_CMD į vienetą .
- Tada resetą nustačius į 1, visi skaitiklio trigeriai nusistato į pradinę poziciją „00000“.

- Resetą pakeitus į 0, skaitiklis galės pradėti savo skaičiavimus.
- Kai skaitiklis pasieks 20 poziciją, savo skaitiklio modulį, įvyks pernaša ir skaitiklio skaičiavimas prasidės iš pradžių(trigeriai : „00000“);

3.4 M1 skaitiklio schemos



4. JM1 skaitiklio realizacija

4.1 M1, M2 skaitiklių vhdl kodai

M1:

```vhdl
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity CNT20 is port (
        CLK         : in std_logic; --Sinchro signalas
        RST         : in std_logic; -- Reset signalas
        CNT_CMD     : in std_logic; -- Komanda
        CNT_C       : out std_logic; --Pernasa
        CNT_O       : out std_logic_vector(4 downto  0) --trigerių skaičius
        );
end CNT20;


architecture rtl of CNT20 is
    signal CNT_A: unsigned (4 downto  0);
begin
    process(CLK, RST, CNT_CMD)
    begin
        if RST = '1' then -- kai RST, reset signalas 1, visi trigeriai nustatomi į 0
            CNT_A <= "00000";
            CNT_C <= '1';
        elsif CLK'event and CLK = '1' and CNT_CMD = '1' then --jeigu reset nėra 1 tada yra patikrinama ar dabartinė padėtis yra mažesnė už pernašą
            if CNT_A < 19  then
                CNT_A <= CNT_A + 1; --Prideda vienetą prie skaitiklio
                if CNT_A = 18 then
                    CNT_C <= '0';
                else
                    CNT_C <= '1';
                end if;
            else
                CNT_C <= '1'; -- jeigu dabartinė padėtis yra pernaša, kitas veiksmas nustato visus trigerius į
                CNT_A <= "00000";
            end if;
        end if;
    end process;
CNT_O <= std_logic_vector(CNT_A);
```

M2:

```vhdl
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
entity CNT41 is port (

        CLK         : in std_logic; --Sinchro signalas
        RST         : in std_logic; -- Reset signalas
        CNT_CMD     : in std_logic; -- Komanda
        CNT_C       : out std_logic; --Pernasa
        CNT_O       : out std_logic_vector(5 downto  0)
        );
end CNT41;


architecture rtl of CNT41 is
    signal CNT_A: unsigned (5 downto  0);
begin
    process(CLK, RST, CNT_CMD)
    begin
        if RST = '1' then -- kai RST, reset signalas 1, visi trigeriai nustatomi į 0
            CNT_A <= "000000";
            CNT_C <= '0';
        elsif CLK'event and CLK = '1' and CNT_CMD = '1' then  --jeigu reset nėra 1 tada yra patikrinama ar dabartinė padėtis yra mažesnė už pernašą
            CNT_A <= CNT_A + 1; --Prideda vienetą prie skaitiklio
            if CNT_A = 41 then
                CNT_C <= '1';
            elsif  CNT_A = 40 then -- jeigu dabartinė padėtis yra pernaša, kitas veiksmas nustato visus trigerius į
                CNT_C <= '1';
                CNT_A <= "000000";
            end if;
        end if;
    end process;
CNT_O <= std_logic_vector(CNT_A);
end rtl;
```

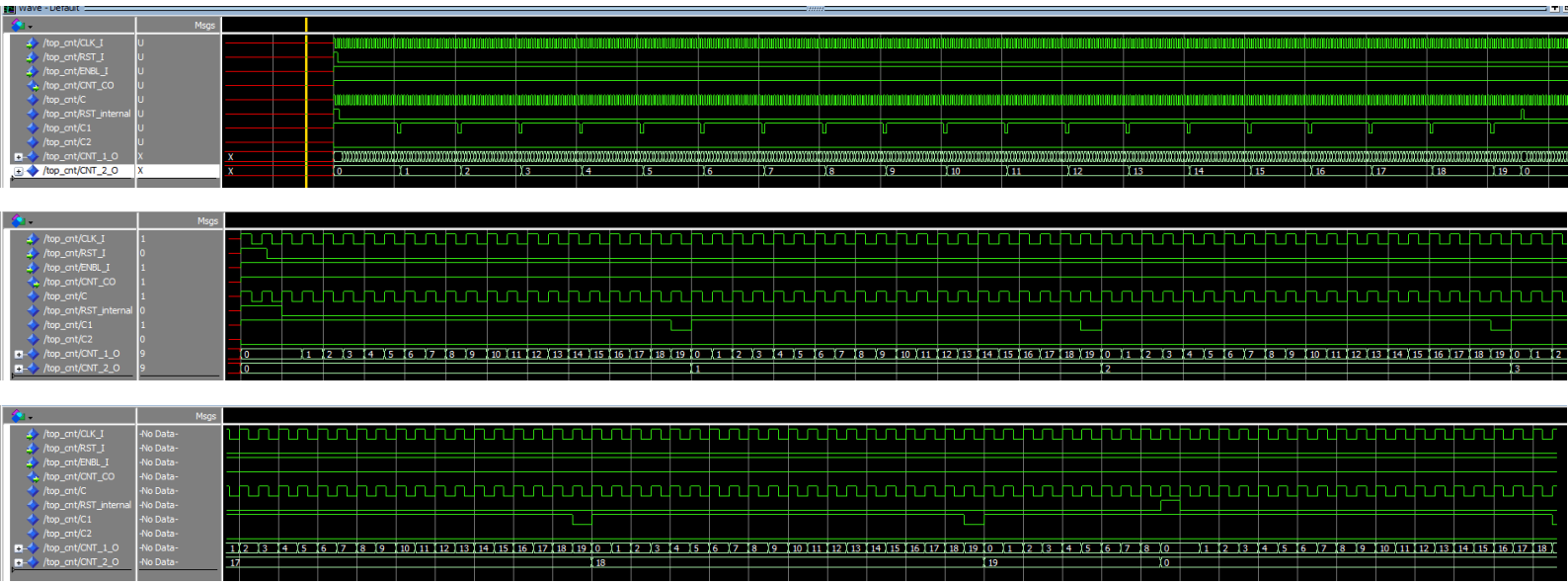
4.2 JM1 skaitiklio vhdl kodas

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity TOP_CNT is port (

        CLK_I   : in std_logic; --Sinchro signalas
        RST_I   : in std_logic; -- Reset signalas
        ENBL_I  : in std_logic; -- Aktyvavimo signalas
        CNT_CO  : out std_logic --Pernasa
        );
end TOP_CNT;

architecture struct of TOP_CNT is

signal C,RST_internal,C1,C2 : std_logic;
signal CNT_1_O :  std_logic_vector(4 downto 0);--trigerių skaičius
signal CNT_2_O :  std_logic_vector(5 downto 0);--trigerių skaičius

component    CNT20
    port    (
        CLK : in std_logic; --Sinchro signalas
        RST : in std_logic; -- Reset signalas
        CNT_CMD  : in std_logic; -- Komanda
        CNT_C    : out std_logic; --Pernasa
        CNT_O    : out std_logic_vector(4 downto  0));
end component;

component    CNT41
    port  (
        CLK : in std_logic; --Sinchro signalas
        RST : in std_logic; -- Reset signalas
        CNT_CMD  : in std_logic; -- Komanda
        CNT_C    : out std_logic; --Kai pasiekia 0
        CNT_O    : out std_logic_vector(5 downto  0) );
end component;


begin
    CNT_1:  CNT20   port map (CLK=>CLK_I,
        RST=>RST_internal, CNT_CMD=>ENBL_I, --Nustatome dydžius
        CNT_C=>C1, CNT_O=>CNT_1_O);
    CNT_2:  CNT41   port map (CLK=> C1,
        RST=>RST_internal, CNT_CMD=>ENBL_I, --Nustatome dydžius
        CNT_C=>C2, CNT_O=>CNT_2_O);
    process(CLK_I,RST_I)
    begin
        if  (RST_I = '1') then -- kai RST_I, reset signalas 1, visi trigeriai nustatomi į 0
            RST_internal <= '1';
        elsif CLK_I'event and CLK_I = '1' then
            if ((CNT_2_O(0) = '1')
            and (CNT_2_O(1) = '1')
            and (CNT_2_O(4) = '1')
            and (CNT_1_O(3) = '1'))           then  -- patikrinama ar nėra pernašos
                RST_internal <= '1';
                CNT_CO <= '1';
            else
                RST_internal <= '0';    -- nustatomi nuliai kai RST_iternal yra vienetas
                CNT_CO <= '0';
            end if;
        end if;
    end process;
end struct;
```

4.3 JM1 skaitiklio testavimo direktyvos

```
force -freeze sim:/top_cnt/CLK_I 1 0, 0 {50 ps} -r 100
force -freeze sim:/top_cnt/C 1 0, 0 {50 ps} -r 100
force -freeze sim:/top_cnt/RST_I 1 0
force -freeze sim:/top_cnt/ENBL_I 1 0
force -freeze sim:/top_cnt/CNT_CO 1 0
run
force -freeze sim:/top_cnt/RST_I 0 0
run
```

4.4 JM1 skaitiklio simuliacija



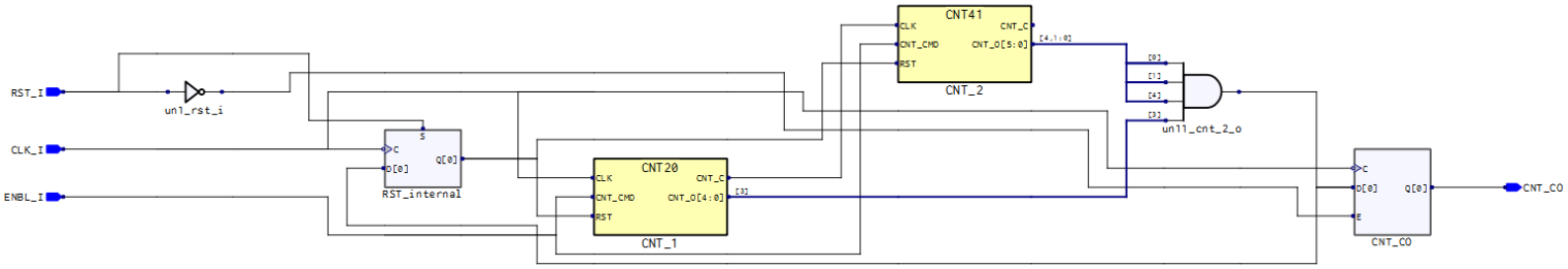
- Pirmiausia, nustatome clock(CLK_I) ir clock(C). Tada CNT_CO į vienetą ir ENBL_I į vienetą .
- Tada resetą nustačius į 1, visi skaitiklio trigeriai nusistato į pradines pozicijas (M1 „00000“, M2“000000“).
- Resetą pakeitus į 0, skaitiklis galės pradėti savo skaičiavimus.
- M1 skaitiklis skaičiuoja iki 20, tada įvyksta pernaša. M1 skaitiklis patampa į 0 („00000“), o M2 pridedamas vienu skaitmeniu. Visa tai vyksta, kol nėra pasiekta šio skaitiklio pernaša, po kurios abu ir M1 ir M2, grįžta į pradinę būseną.

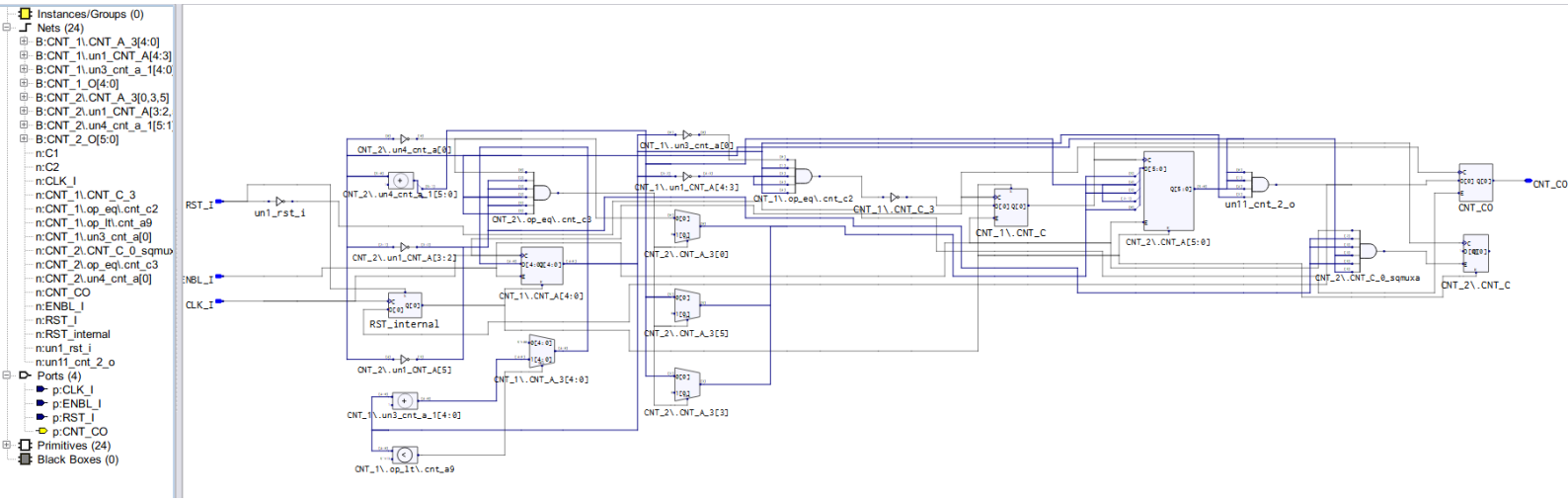
$$S2 = (JM - 1) div\ M1 = \frac{388}{20} = 19 \text{ (antrojo skaitiklio būsena)}$$

$$S1 = (JM - 1) mod\ M1 = 388 mod\ 20 = 8 \text{ (pirmojo skaitiklio būsena)}$$

- Taip skaičiuojat gaunasi, jog 20*19 + 9 = 389 (JM skaičius).

4.5 JM1 skaitiklio schemos





5. JM2 skaitiklio realizacija

5.1 M3 skaitiklio vhdl kodas

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity CNT150 is port (

        CLK         : in std_logic; --Sinchro signalas
        RST         : in std_logic; -- Reset signalas
        CNT_CMD     : in std_logic; -- Komanda
        CNT_C       : out std_logic; --Pernasa
        CNT_O       : out std_logic_vector(7 downto  0)--trigerių skaičius
        );
end CNT150;


architecture rtl of CNT150 is
    signal CNT_A: unsigned (7 downto  0);
begin
    process(CLK, RST, CNT_CMD)
    begin
        if RST = '1' then-- kai RST, reset signalas 1, visi trigeriai nustatomi į 0
            CNT_A <= "00000000";
            CNT_C <= '0';
        elsif CLK'event and CLK = '1' and CNT_CMD = '1' then --jeigu reset nėra 1 tada yra patikrinama ar dabartinė padėtis yra mažesnė už pernašą
            CNT_A <= CNT_A + 1;--Prideda vienetą prie skaitiklio
            if CNT_A = 150 then
                CNT_C <= '1';
            elsif  CNT_A = 149 then     -- jeigu dabartinė padėtis yra pernaša, kitas veiksmas nustato visus trigerius į
                CNT_C <= '1';
                CNT_A <= "00000000";
            end if;
        end if;
    end process;
CNT_O <= std_logic_vector(CNT_A);
end rtl;
```

6. M1 skaitiklio vhdl kodo pritaikymas PLIS matricai:

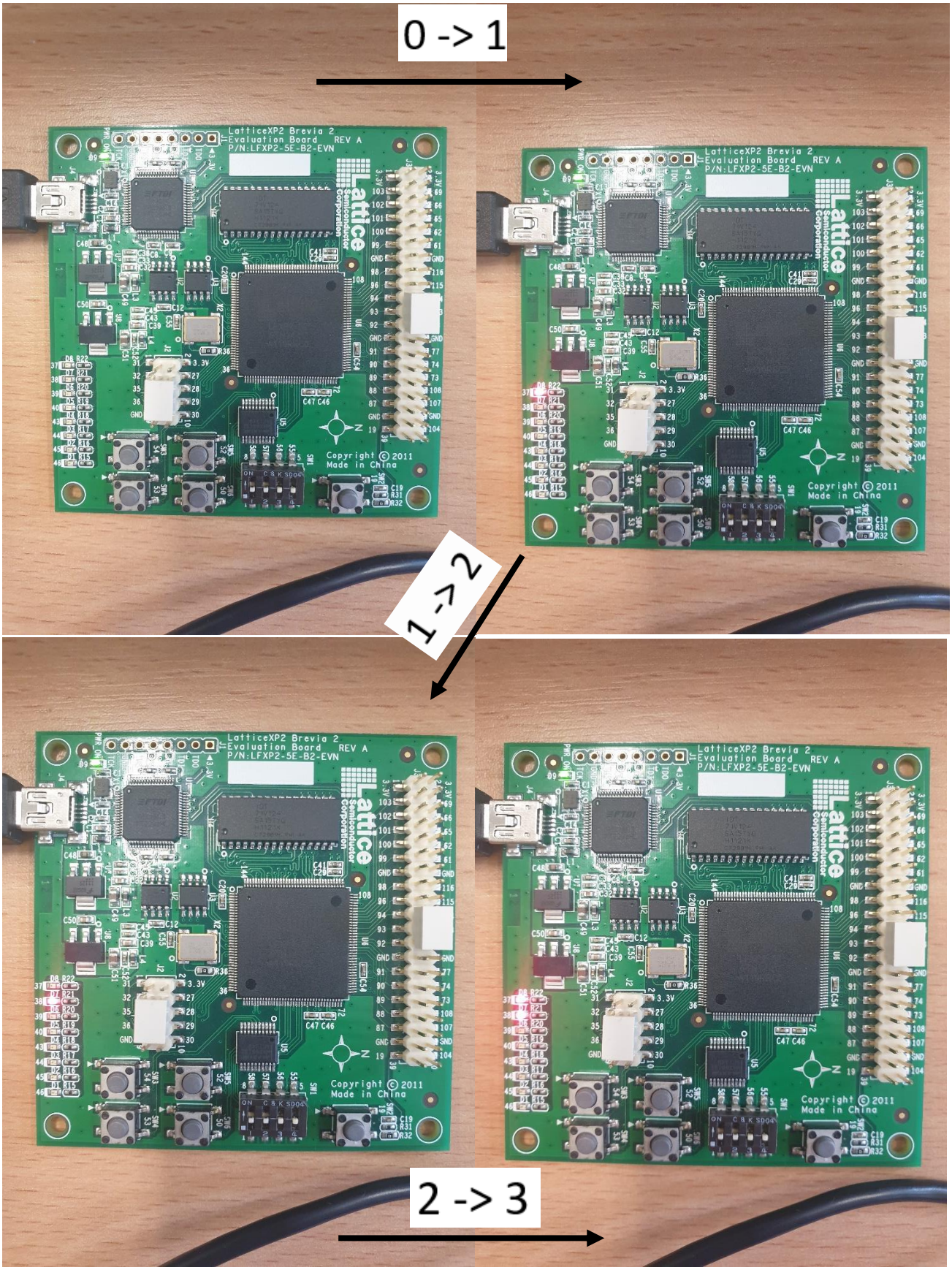
6.1 M1 vhdl kodas

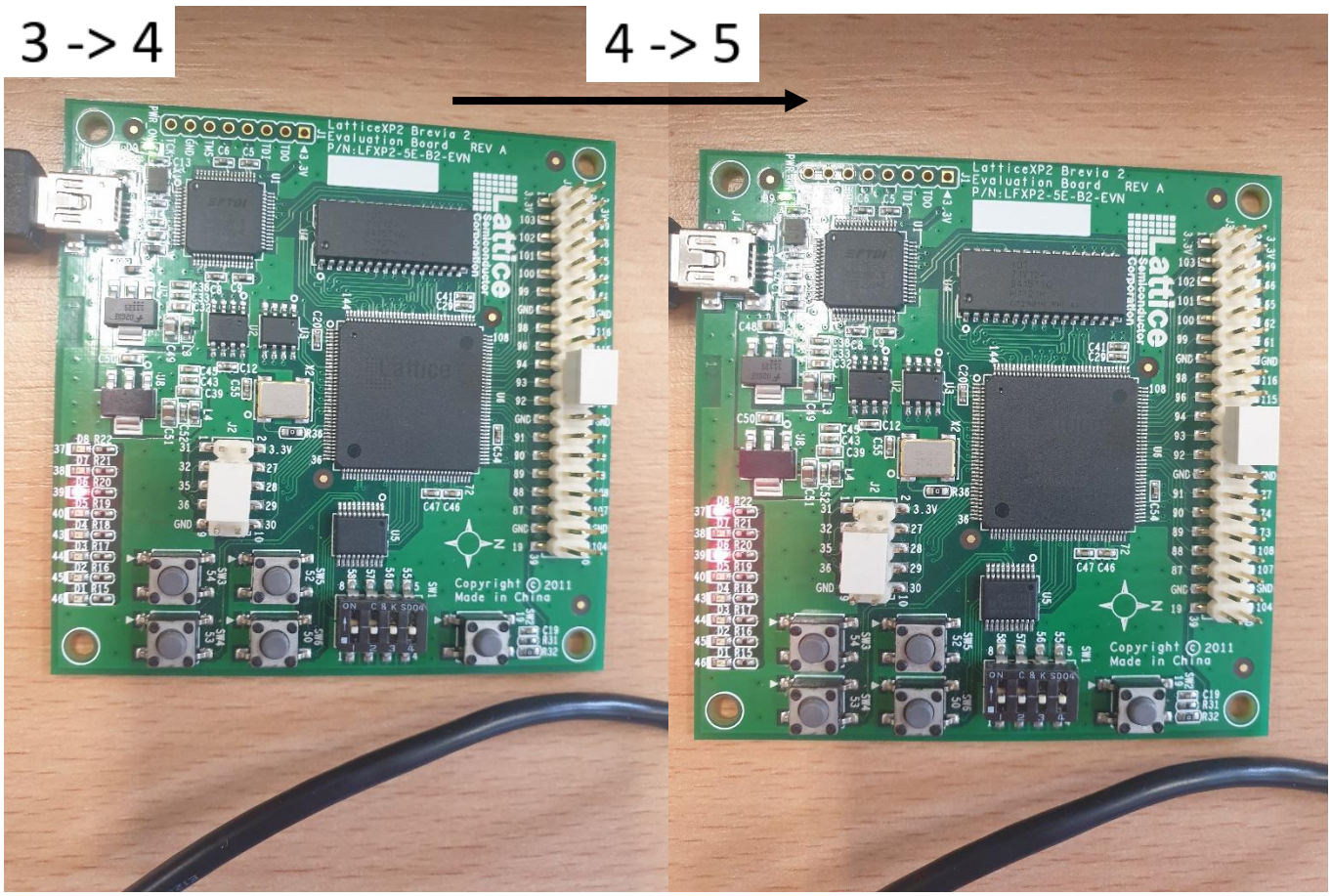
```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity CNT20 is port (
        CLK         : in std_logic; --Sinchro signalas
        RST         : in std_logic; -- Reset signalas
        CNT_CMD     : in std_logic; -- Komanda
        CNT_C       : out std_logic; --Pernasa
        CNT_O       : out std_logic_vector(4 downto  0) --trigerių skaičius
        );
end CNT20;


architecture rtl of CNT20 is
    signal CNT_A: unsigned (4 downto  0);
begin
    process(CLK, RST, CNT_CMD)
    begin
        if RST = '0' then -- kai RST, reset signalas 0, visi trigeriai nustatomi į 0
            CNT_A <= "00000";
            CNT_C <= '1';
        elsif CLK'event and CLK = '1' and CNT_CMD = '1' then --jeigu reset nėra 1 tada yra patikrinama ar dabartinė padėtis yra mažesnė už pernašą
            if CNT_A < 19  then
                CNT_A <= CNT_A + 1; --Prideda vienetą prie skaitiklio
                if CNT_A = 18 then
                    CNT_C <= '0';
                else
                    CNT_C <= '1';
                end if;
            else
                CNT_C <= '1'; -- jeigu dabartinė padėtis yra pernaša, kitas veiksmas nustato visus trigerius į
                CNT_A <= "00000";
            end if;
        end if;
    end process;
CNT_O <= not(std_logic_vector(CNT_A));

end rtl;
```

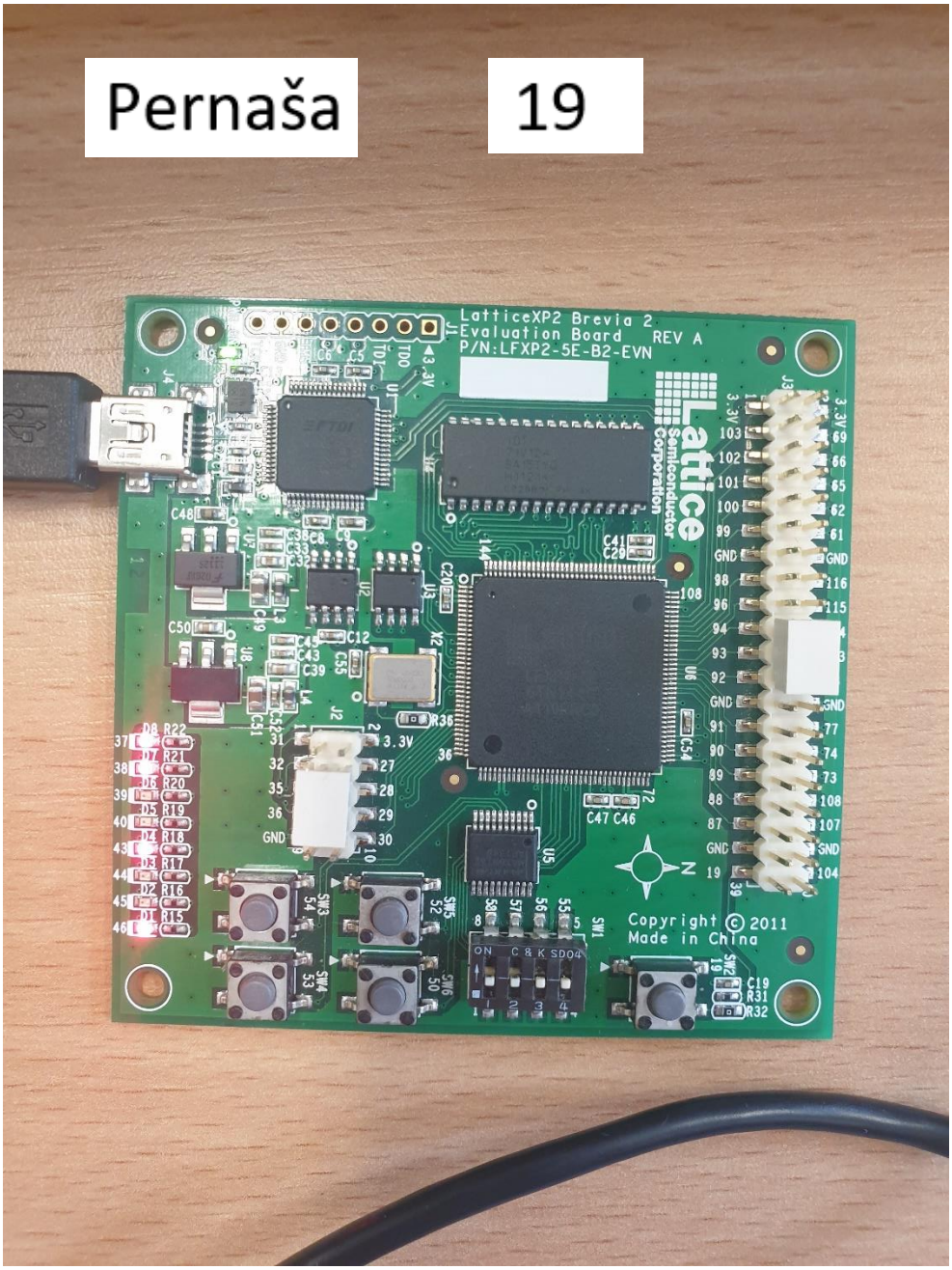
6.2 Fizinių kontaktų lentelė

	Name	Group By	Pin	BANK	VREF	IO_TYPE	PULLMODE	DRIVE	SLEWRATE	PCICLAMP	OPENDRAIN	Outload (pF)	MaxSkew	Clock Load Only	SwitchingID	Ground plane PCB noise (mV)
1	All Ports	N/A	N/A	N/A	N/A			N/A				N/A	N/A	N/A		0.00
1.1	Input	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
1.1.1	CNT_CMD	N/A	56(56)	5(5)	N/A	LVCMOS2...	UP(UP)	NA(NA)	FAST(FAST)	OFF(OFF)	OFF(OFF)	N/A		N/A	N/A	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	N/A	N/A	N/A	N/A
1.1.2.1	CLK	N/A	52(52)	5(5)	N/A	LVCMOS2...	UP(UP)	NA(NA)	FAST(FAST)	OFF(OFF)	OFF(OFF)	N/A		N/A	N/A	N/A
1.1.3	RST	N/A	58(58)	5(5)	N/A	LVCMOS2...	UP(UP)	NA(NA)	FAST(FAST)	OFF(OFF)	OFF(OFF)	N/A		N/A	N/A	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	N/A	N/A	N/A	N/A
1.2.1	CNT_C	N/A	46(46)	5(5)	N/A	LVCMOS2...	UP(UP)	12(12)	FAST(FAST)	OFF(OFF)	OFF(OFF)	0.000		N/A		0.00
1.2.2	CNT_O[0]	N/A	37(37)	5(5)	N/A	LVCMOS2...	UP(UP)	12(12)	FAST(FAST)	OFF(OFF)	OFF(OFF)	0.000		N/A		0.00
1.2.3	CNT_O[1]	N/A	38(38)	5(5)	N/A	LVCMOS2...	UP(UP)	12(12)	FAST(FAST)	OFF(OFF)	OFF(OFF)	0.000		N/A		0.00
1.2.4	CNT_O[2]	N/A	39(39)	5(5)	N/A	LVCMOS2...	UP(UP)	12(12)	FAST(FAST)	OFF(OFF)	OFF(OFF)	0.000		N/A		0.00
1.2.5	CNT_O[3]	N/A	40(40)	5(5)	N/A	LVCMOS2...	UP(UP)	12(12)	FAST(FAST)	OFF(OFF)	OFF(OFF)	0.000		N/A		0.00
1.2.6	CNT_O[4]	N/A	43(43)	5(5)	N/A	LVCMOS2...	UP(UP)	12(12)	FAST(FAST)	OFF(OFF)	OFF(OFF)	0.000		N/A		0.00

6.3 M1 skaitiklio realizacija


0 -> 1
1 -> 2
2 -> 3
LatticeXP2 Brevia 2
Evaluation Board REV A
P/N:LFXP2-5E-B2-EVN
Lattice
Semiconductor
Corporation
Copyright © 2011
Made in China


3 -> 4
4 -> 5
LatticeXP2 Brevia 2
Evaluation Board REV A
P/N:LFXP2-5E-B2-EVN
Lattice
Semiconductor
Corporation
Copyright © 2011
Made in China



7. Išvados

- Išmokau skaitiklių veikimo principus.
- Išmokau užrašyti kodą VHDL kalboje.
- Išmokau naudojant VHDL kodą suprojektuoti schemas naudojant „Synplify pro“
- Išmokau sujungti kelis skaitiklius į vieną.
- Išmokau atvaizduoti skaitiklį simuliacijose.