

**Kauno technologijos universitetas**

Informatikos fakultetas

**Skaitmeninės logikos pradmenys P175B100**

Ketvirtojo laboratorinio darbo ataskaita

**Studentas**

Lukas Kuzmickas

**Dėstytoja**

Rasa Brūzgienė

**Kaunas 2022**

# Turinys

<b>1. Įvadas .....</b>	<b>3</b>
<b>1.1 Individuali darbo užduotis – 121 numeris. ....</b>	<b>3</b>
<b>2. Pagrindinė dalis .....</b>	<b>5</b>
<b>2.1 M1 skaitiklio vhdl kodas (su komentarais). ....</b>	<b>5</b>
<b>2.2 M2 skaitiklio vhdl kodas (su komentarais). ....</b>	<b>8</b>
<b>2.3 JM1 skaitiklio vhdl kodas (su komentarais). ....</b>	<b>11</b>
<b>2.4 JM1-300 skaitiklio testavimo direktyvos. ....</b>	<b>15</b>
<b>2.5 JM1 skaitiklio simuliacijos rezultatai. ....</b>	<b>15</b>
<b>2.6 JM1 skaitiklio sugeneruotos RTL schemas .....</b>	<b>16</b>
<b>2.7 M1 skaitiklio vhdl kodo pritaikymas – Lattice Brevia PLIS matricai. ....</b>	<b>16</b>
<b>2.7.1 Pakoreguotas M1 skaitiklio VHDL kodas (su komentarais). ....</b>	<b>17</b>
<b>2.7.2 Priskirti fiziniai kontaktai PLIS matricai. ....</b>	<b>19</b>
<b>2.7.3 M1 skaitiklio PLIS matricos realizacijos nuotraukos. ....</b>	<b>20</b>
<b>3. Išvados .....</b>	<b>22</b>

# 1. Įvadas

## 1.1 Individuali darbo užduotis – 121 numeris.

Ketvirtojo laboratorinio darbo individuali užduotis:



1 pav. Ketvirtojo laboratorinio darbo individuali užduotis

Pagal šiuos individualios užduoties sąlygas žinome, kad mums reikia sumodeliuoti:

- M1 skaitiklį (sumatorių) – kuris sumuoja iki 13. (nuo 0 iki 12).
- M2 skaitiklį (sumatorių) – kuris sumuoja iki 44. (nuo 0 iki 43).
- M3 skaitiklį (sumatorių) – kuris sumuoja iki 11. (nuo 0 iki 10).
- JM1 jungtinį skaitiklį (sumatorių), kuris yra sudarytas iš M1 ir M2 skaitiklių, kuris bendrai sumuoja iki 300.
- JM2 jungtinį skaitiklį (sumatorių), kuris yra sudarytas iš M1, M2 ir M3 skaitiklių, kuris bendrai sumuoja iki 967.
- M1 arba M2 skaitiklių PLIS Lattice Brevia matricos realizacija.

Formulės:

$$JM1 = \lfloor \frac{JM1}{M1} \rfloor * M1 + (JM1 \bmod M2)$$

2 pav. Bendro skaitiklio apskaičiavimo formulė

$$M2_{FIN} = \lfloor \frac{JM1}{M1} \rfloor$$

3 pav. M2 skaitiklio vidinės reset reikšmės sugeneravimo formulė

Skaičiavimai:

JM1 skaitiklio realizacijai, apsiskaičiuojame, kada turi būti sudarytas vidinis reset reikšmės sugeneravimo signalas M1 ir M2 skaitikliam.

Iš 3 paveikslėlio formulės galime apsiskaičiuoti, kada turi būti sugeneruotas M2 skaitiklio reset signalas:

$$M2_{fin} = \frac{300}{13} \approx 23$$

M2 skaitiklio reset signalas turi būti suformuotas, kai M2 skaitiklis pasiekia 23.

Iš 2 paveikslėlio formulės galime apsiskaičiuoti, kada turi būti sugeneruotas M1 skaitiklio reset signalas:

$$JM1 = 13 * 23 + M1Fin = 300$$

M1Fin reikšmė bus lygi 1, tai M1 skaitiklis sugeneruoja reset signalą, kai pasiekia 0. (skaičiuojam nuo 0 iki  $\infty$ ).

Tai maksimali reikšmė bus  $JM1 = 23 * 13 + 1 = 300$ .

Konvertuojame skaičių 23 iš dešimtainio į dvejetainį kodą.

$$23_{10} = 10111_2$$

## 2. Pagrindinė dalis

### 2.1 M1 skaitiklio vhdL kodas (su komentarais).

Sudarome M1 skaitiklio VHDL kodą, kuris sumuoja iki 13.

Pradžioje pridedame mums reikalingas bibliotekas, kurias naudosime.

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

4 pav. VHDL kodas

Vėliau apsirašome įėjimus ir išėjimus šito skaitiklio, kur:

- CLK – skaičiavimo signalas.
- RST – reset signalas.
- CNT-CMD – signalas, kuris leidžia skaitikliui dirbti.
- CNC\_C – pernašos signalas.
- CNT\_O – skaitiklio išvestys.

```

entity CNT13 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(3 downto 0)
);
end CNT13;

```

5 pav. VHDL kodas

Mūsų M1 skaitiklis sumuoja iki 13, todėl apsiskaičiuojame, kad mums reikės  $2^4$  ( $16 > 13$ ) reikšmių arba 16 reikšmių – 4 bitų.

```

architecture rtl of CNT13 is
    signal CNT_A: unsigned (3 downto 0);
begin
    process(CLK, RST, CNT_CMD)
    begin
        if RST = '1' then
            CNT_A <= "0000";
            CNT_C <= '1';
        elsif CLK'event and CLK = '1' and CNT_CMD = '1' then

```

6 pav. VHDL kodas

Mūsų modeliuojamas skaitiklis sumuoja nuo 0 iki 12 (13 reikšmių). Jis sumuoja toliau iki kol pasiekia 11 reikšmę, tada suformuojamas pernašos signalas CNT\_C. Kai skaitiklis pasiekia maksimalią reikšmę – 12, yra suformuojamas reset signalas, kuris nustato skaitiklį į pradinę būseną (0000) ir vėl kartojamas skaitiklio darbas.

```
if CNT_A < 12 then
    CNT_A <= CNT_A + 1;
    if CNT_A = 11 then
        CNT_C <= '0';
    else
        CNT_C <= '1';
    end if;
else
    CNT_C <= '1';
    CNT_A <= "0000";
end if;

end if;

end process;

CNT_O <= std_logic_vector(CNT_A);

end rtl;
```

7 pav. VHDL kodas

Operatorius CNT-O atiduoda mums skaitiklio rezultatus - dvejetainį vektorių.

Taip sumodeliuojamas CNT\_13 skaitiklis, kuris geba sumuoti iki 13, VHDL kodu.

## 2.2 M2 skaitiklio vhd1 kodas (su komentarais).

Sudarome M2 skaitiklio VHDL kodą, kuris sumuoja iki 44.

Pradžioje pridedame mums reikalingas bibliotekas, kurias naudosime.

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

8 pav. VHDL kodas

Po to apsirašome įėjimus ir išėjimus šito skaitiklio, kur:

- CLK – skaičiavimo signalas.
- RST – reset signalas.
- CNT-CMD – signalas, kuris leidžia skaitikliui dirbti.
- CNC\_C – pernašos signalas.
- CNT\_O – skaitiklio išvestys.



entity CNT44 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 CNT44;

9pav. VHDL kodas

Mūsų M1 skaitiklis sumuoja iki 44, todėl apsiskaičiuojame, kad mums reikės  $2^6$  ( $64 > 44$ ) reikšmių arba 64 reikšmių – 6 bitų.

architecture rtl of CNT44 is

```
    signal CNT_A: unsigned (5 downto 0);
```

10 pav. VHDL kodas

Mūsų modeliuojamas skaitiklis sumuoja nuo 0 iki 43 (44 reikšmės). Jis sumuoja toliau iki kol pasiekia 42 reikšmę, tada suformuojamas pernašos signalas CNT\_C. Kai skaitiklis pasiekia maksimalią reikšmę – 43, yra suformuojamas reset signalas, kuris nustato skaitiklį į pradinę būseną (000000) ir vėl kartojamas skaitiklio darbas.

```
if CNT_A < 43 then
    CNT_A <= CNT_A + 1;
    if CNT_A = 42 then
        CNT_C <= '0';
    else
        CNT_C <= '1';
    end if;
else
    CNT_C <= '1';
    CNT_A <= "000000";
end if;

end if;

end process;

CNT_O <= std_logic_vector(CNT_A);

end rtl;
```

11 pav. VHDL kodas

Operatorius CNT-O atiduoda mums skaitiklio išvestis - dvejetainį vektorių.

Taip sumodeliuojamas CNT\_44 skaitiklis, kuris geba sumuoti iki 44, VHDL kodu.

### 2.3 JM1 skaitiklio vhd1 kodas (su komentarais).

Turint M1 ir M2 skaitiklius, galime sudaryti JM1 jungtinį skaitiklį, kuris bendrai sumuoja iki 300. Tam iš pradžių susiskaiciavome, kad M2 skaitiklis suformuoja reset signalą, kai pasiekia 23 reikšmę, o M1, kai pasiekia 0.

Iš pradžių aprašome reikalingas bibliotekas, kurias naudosime.

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

12 pav. VHDL kodas

Po to apsirašome JM1\_cntr300 jungtinio skaitiklio įėjimus ir išėjimus šito skaitiklio, kur:

- CLK\_I – skaičiavimo signalas.
- RST\_I – reset signalas.
- ENBL\_I – signalas, kuris leidžia skaitikliui dirbti.

- CNC\_CO – pernašos signalas.

```
entity JM1_cntr300 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 JM1_cntr300;
```

13 pav. VHDL kodas

Tuo pačiu įtraukiame M1 ir M2 skaitiklius ir aprašome juos.

```
architecture struct of JM1_cntr300 is

    signal C,RST_internal,C1,C2 : std_logic;

    signal CNT_1_O : std_logic_vector(3 downto 0);

    signal CNT_2_O : std_logic_vector(5 downto 0);|
```

14 pav. VHDL kodas

Aprašome M1 skaitiklį.

```
component    CNT13

    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(3 downto 0));

end component;
```

15 pav. VHDL kodas

Aprašome M2 skaitiklį.

```
component    CNT44

    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;
```

16 pav. VHDL kodas

Šiems skaitikliams priskiriam JM1 skaitiklio įvestis ir išvestis.

```
begin

    CNT_1:    CNT13    port map (CLK=>CLK_I,

                                RST=>RST_internal, CNT_CMD=>ENBL_I,

                                CNT_C=>C1, CNT_O=>CNT_1_O);

    CNT_2:    CNT44 port map (CLK=> C1,

                                RST=>RST_internal, CNT_CMD=>ENBL_I,

                                CNT_C=>C2, CNT_O=>CNT_2_O);
```

17 pav. VHDL kodas

Pagal užduotį apsirašome, kada privalo būti suformuotas vidinis RST\_internal signalas – M2 (kai pasiekia 23) ir M1 (kai pasiekia 0). Tai apsirašome: CNT\_2\_O (10111) ir CNT\_1\_O (0). Kai pasiekiamos šios reikšmės skaitiklis sugrįžta į savo pradinę būseną ir pradeda savo darbą iš naujo.

```
if ((CNT_2_O(4) = '1')
    and (CNT_2_O(2) = '1')
    and (CNT_2_O(1) = '1')
    and (CNT_2_O(0) = '1')
    and (CNT_1_O(0) = '0')) then

    RST_internal <= '1';
    CNT_CO <= '1';

else

    RST_internal <= '0';
    CNT_CO <= '0';

end if;

end if;

end process;

end struct;
```

18 pav. VHDL kodas

Taip sudaromas jungtinis JM1-300 skaitiklis, kuris sugeba bendrai sumuoti iki 300.

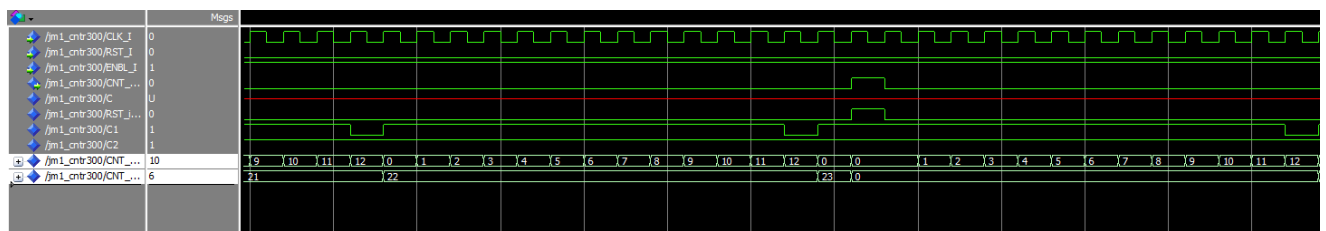
## 2.4 JM1-300 skaitiklio testavimo direktyvos.

Sudarome jungtinio skaitiklio simuliacijos direktyvas.

```
force -freeze sim:/jm1_cntr300/CLK_I 0 0, 1 {10 ps} -r 20
force -freeze sim:/jm1_cntr300/ENBL_I 1 0, 0 {500 ps}
force -freeze sim:/jm1_cntr300/ENBL_I 1 560
force -freeze sim:/jm1_cntr300/RST_I 1 0, 0 {5 ps}
run 50000
```

19 pav. JM1-300 skaitiklio direktyvos

## 2.5 JM1 skaitiklio simuliacijos rezultatai.

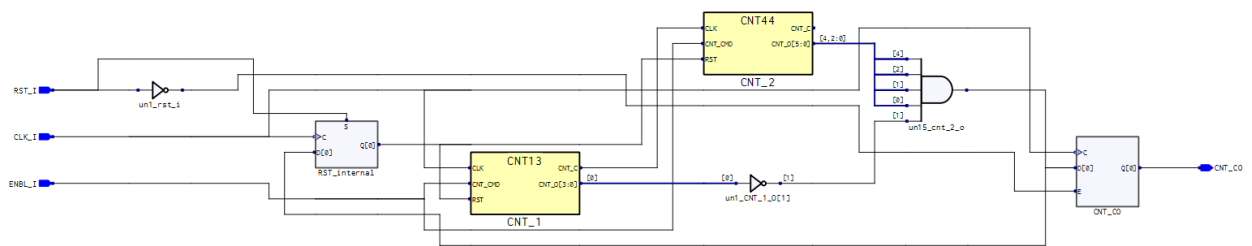


20 pav. JM1-300 jungtinio skaitiklio rezultatai atliktos simuliacijos.

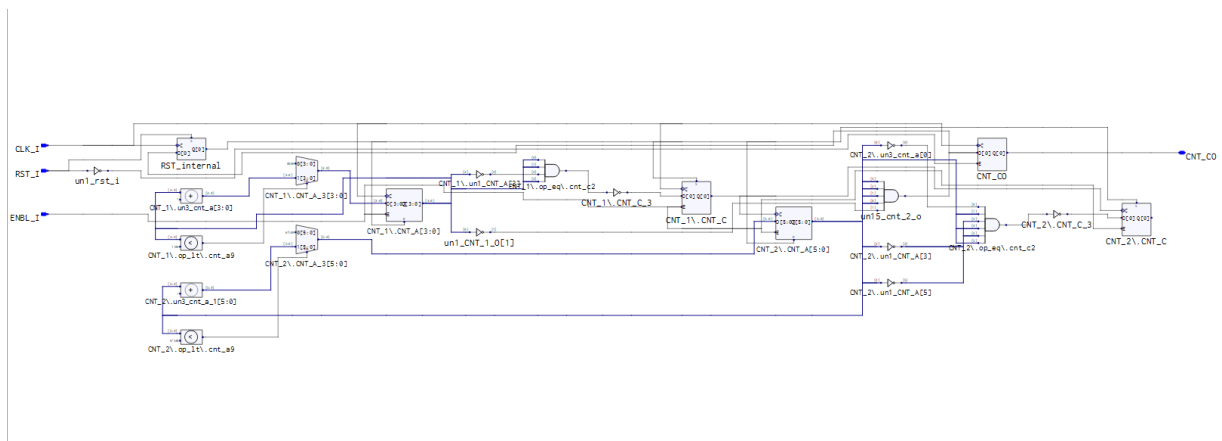
Iš simuliacijos matome, kad skaitiklis sumuoja iki kol pasiekia M2 skaitiklio (23 reikšmę), kur sudaromas pernašos signalas ir viskas nustatoma į pradinę padėtį. Apskaičiuojame ar viskas veikia korektiškai –  $13 \cdot 23 + 1 = 300$ . Mūsų JM1 jungtinis skaitiklis bendrai sumuoja iki 300.

## 2.6 JM1 skaitiklio sugeneruotos RTL schemos

Pasinaudodami **Tools** => **Simplify Pro for Lattice** ir pasirinkdami **HDL-Analyst** => **RTL**, galime sugeneruoti JM1 jungtinio skaitiklio schemas: "Hierarchical view" ir "Flattened view".



21 pav. JM1 RTL-Hierarchical view



22 pav. JM1 RTL- Flattened view

## 2.7 M1 skaitiklio vhd1 kodo pritaikymas – Lattice Brevia PLIS matricai.

Toliau mūsų užduotis reikalauja M1 arba M2 skaitiklį pritaikyti PLIS matricai. PLIS matricos realizacijai, dėl skaičių kiekio buvo pasirinktas M1 skaitiklis.



### 2.7.1 Pakoreguotas M1 skaitiklio VHDL kodas (su komentarais).

Viskas apsidrašo, kaip ir M1 skaitiklio VHDL kode, bet yra keli esminiai skirtumai.

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

23 pav. VHDL kodas

Pridedame skaitiklio įėjimus ir išėjimus.

```
entity CNT13 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(3 downto 0)  
  
);  
  
end CNT13;
```

24 pav. VHDL kodas

Nustatome bitų skaičių.

architecture rtl of CNT13 is

signal CNT\_A: unsigned (3 downto 0);

25 pav. VHDL kodas

**Norint realizuoti mūsų M1 skaitiklį PLIS matricai, reikia invertuoti mūsų RST signalo sąlygą.**

---

if RST = '0' then

CNT\_A <= "0000";

CNT\_C <= '1';

elsif CLK'event and CLK = '1' and CNT\_CMD = '1' then

if CNT\_A < 12 then

CNT\_A <= CNT\_A + 1;

if CNT\_A = 11 then

CNT\_C <= '0';

else

CNT\_C <= '1';

end if;

else

CNT\_C <= '1';

CNT\_A <= "0000";

end if;

end if;

26 pav. VHDL kodas

Tuo pačiu reikia invertuoti ir mūsų CNT\_O išvedimo signalus (not), nes jie bus prijungiami prie LED kontaktų.

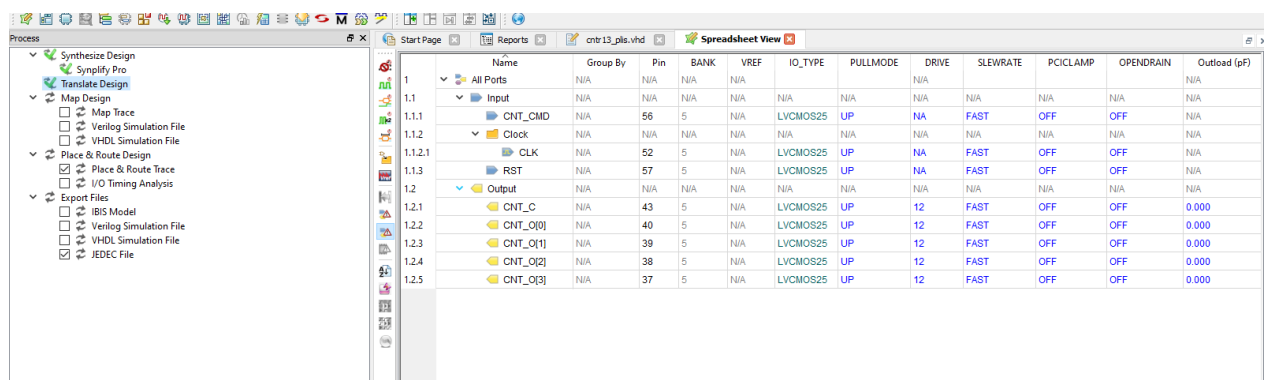
```
CNT_O <= not(std_logic_vector(CNT_A));

end rtl;
```

27 pav. VHDL kodas

## 2.7.2 Priskirti fiziniai kontaktai PLIS matricai.

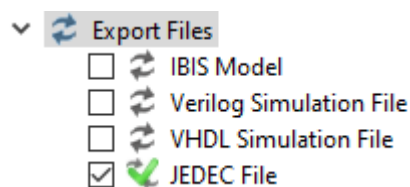
Pasinaudodami „SpreadSheetView“ įrankiu priskiriame atitinkamus kontaktus mūsų M1 skaitiklio matricai.



	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
1.1	Input	N/A	N/A	N/A	N/A							N/A
1.1.1	CNT_CMD	N/A	56	5	N/A	LVC MOS25	UP	NA	FAST	OFF	OFF	N/A
1.1.2	Clock	N/A	N/A	N/A	N/A							N/A
1.1.2.1	CLK	N/A	52	5	N/A	LVC MOS25	UP	NA	FAST	OFF	OFF	N/A
1.1.3	RST	N/A	57	5	N/A	LVC MOS25	UP	NA	FAST	OFF	OFF	N/A
1.2	Output	N/A	N/A	N/A	N/A							N/A
1.2.1	CNT_C	N/A	43	5	N/A	LVC MOS25	UP	12	FAST	OFF	OFF	0.000
1.2.2	CNT_O[0]	N/A	40	5	N/A	LVC MOS25	UP	12	FAST	OFF	OFF	0.000
1.2.3	CNT_O[1]	N/A	39	5	N/A	LVC MOS25	UP	12	FAST	OFF	OFF	0.000
1.2.4	CNT_O[2]	N/A	38	5	N/A	LVC MOS25	UP	12	FAST	OFF	OFF	0.000
1.2.5	CNT_O[3]	N/A	37	5	N/A	LVC MOS25	UP	12	FAST	OFF	OFF	0.000

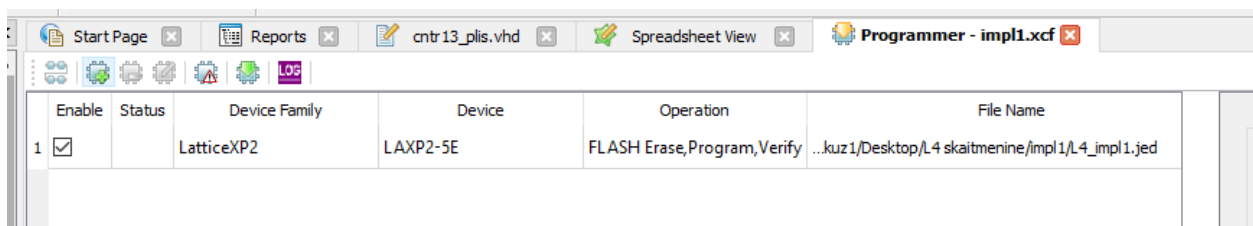
28 pav. Kontaktų priskirimas matricai

Vėliau pasinaudodami Export Files funkcija, galime sugeneruoti „JEDEC“ failus, kurie bus naudojami užprogramuoti PLIS matricai.



29 pav. „JEDEC“ file sukūrimas

Pasinaudodami **Tools=>Programmer** pasirenkame mūsų matricos USB jungtį ir „JEDEC“ sugeneruotą failą ir naudodami **Design => Program** užprogramuojame mūsų PLIS matricą.

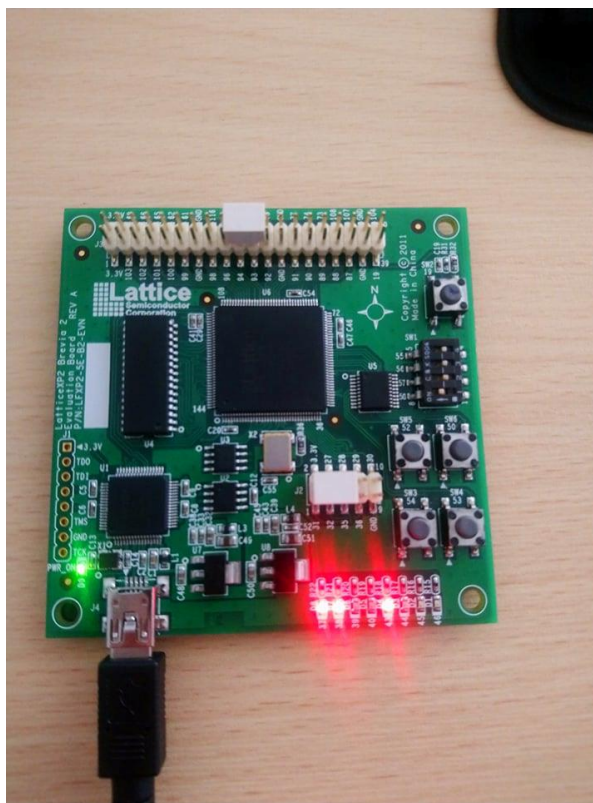


30 pav. Programmer ir Program Tools

### 2.7.3 M1 skaitiklio PLIS matricos realizacijos nuotraukos.

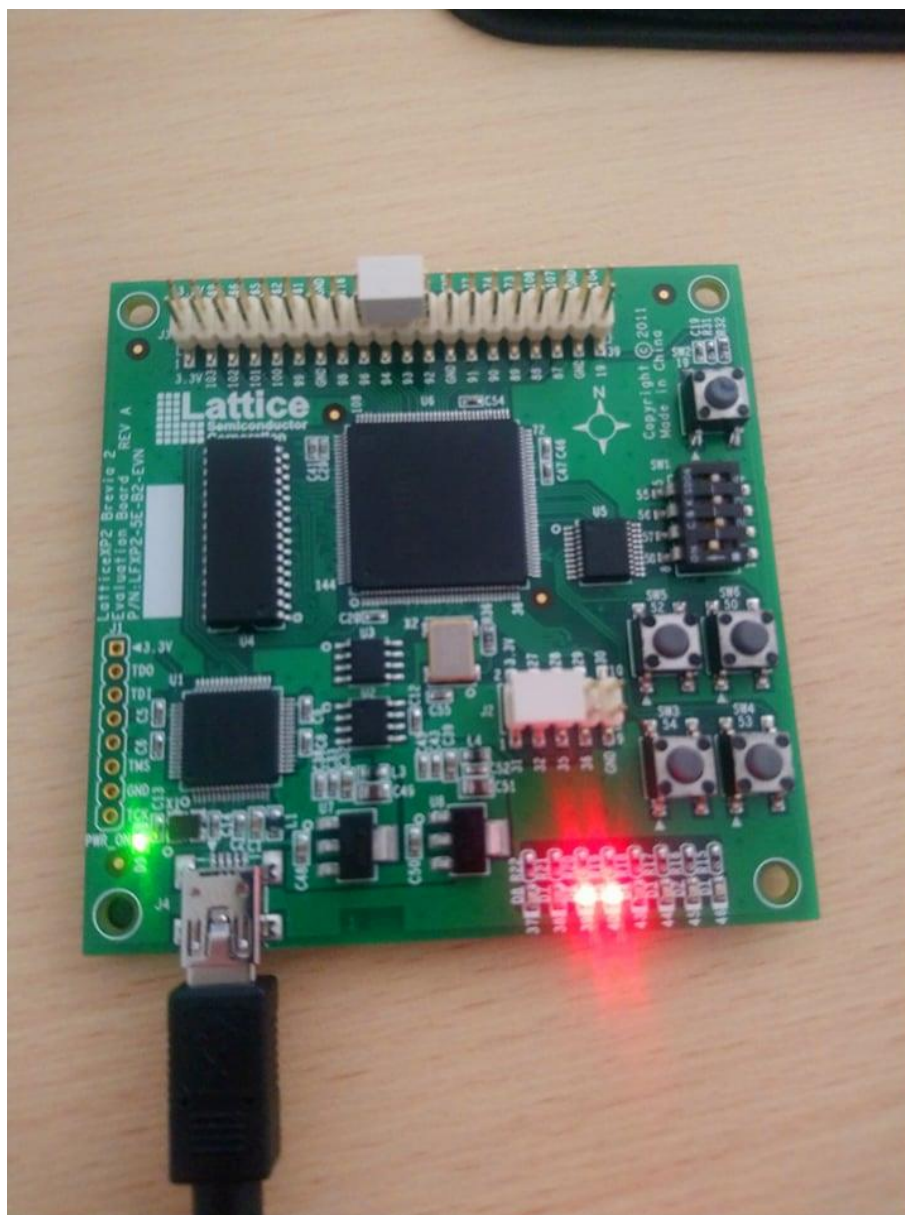
Užprogramuota PLIS matrica testuojame priskirdami tam tikriems mūsų kontaktams tinkamas reikšmes ir keisdami CLOCK mygtuko pagalba, stebime mūsų M1 skaitiklio darbą.

Keičiant CLOCK reikšmes sumuojama reikšmė, iki kol pasiekiamas 1100 (12) ir suformuojamas pernašos signalas ir viskas yra nustatoma į pradinę būseną.



31 pav. Lattice Brevia M1 skaitiklio pernašos signalas

Dar vienam bandymui pateiktas sumavimo pavyzdys – 0011 (3).



32 pav. Lattice Brevia M1 skaitiklio realizacija

### 3. Išvados

- Sėkmingai sumodeliavome M1 skaitiklį, kuris sugeba sumuoti iki 13.
- Sėkmingai sumodeliavome M2 skaitiklį, kuris sugeba sumuoti iki 44.
- Sėkmingai realizavome jungtinį JM1 skaitiklį, sudaryta iš M1 ir M2 skaitiklių, kuris sugeba bendrai sumuoti iki 300.
- Pritaikėme PLIS matricos realizaciją M1 skaitikliui.
- Pagiliname žinias apie skaitiklius ir jo mikrooperacijos veikimą.
- Praktinė darbo patirtis su PLIS matricomis – Lattice Diamond platformoje.