

**Kauno technologijos universitetas**

Informatikos fakultetas

Schemų aprašymas VHDL kalba (skaitikliai)

P175B100 Skaitmeninės logikos pradmenų ketvirtas laboratorinis darbas

**Projekto autorius**

Gustas Klevinskas

**Akademinė grupė**

IFF-8/7

**Vadovai**

Doc. Tomas Adomkus

Kaunas, 2019

Turinys

[Įvadas 3](#_Toc8327602)

[Skaitikliai 4](#_Toc8327603)

[M1 4](#_Toc8327604)

[M2 5](#_Toc8327605)

[M3 6](#_Toc8327606)

[JM2 7](#_Toc8327607)

[Rezultatai 9](#_Toc8327608)

[Perėjimas prie FPGA 10](#_Toc8327609)

[Išvados 11](#_Toc8327610)

# Įvadas

Darbo tikslas – naudojantis VHDL kalba suprojektuoti skaitiklį, kuris naudodamas tris mažesnius skaitiklius, galėtų skaičiuoti iki duotosios vertės. Patikrinti skaitiklio veikimą programuojamos logikos schemoje.

Užduotys:

1. Pagal duotus reikalavimus aprašyti M1, M2 ir M3 skaitiklius VHDL kalba;
2. Sukurti papildomą failą, kuris apjungtų M1, M2 ir M3 skaitiklius;
3. Parašyti stimulą ir patikrinti suprojektuoto skaitiklio funkcionavimą;
4. Pakoreguoti schemą ir ją įkelti į FPGA plokštę.

# Skaitikliai

## M1

Šis skaitiklis turi skaičiuoti moduliu 8. Kadangi maksimali skaitiklio vertė bus 7, jam suprojektuoti užteks 3 bitų.

Skaitiklio M1 VHDL failas:

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.numeric\_std.all;

entity M1 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 (2 downto 0));

end M1;

architecture rtl of M1 is

signal CNT\_A : unsigned (2 downto 0);

begin

process (CLK, RST, CNT\_CMD)

begin

if RST = '1' then

CNT\_A <= "000";

CNT\_C <= '1';

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

if CNT\_A < 7 then -- Pernasos signalas

CNT\_A <= CNT\_A + 1;

if CNT\_A = 6 then

CNT\_C <= '0';

else

CNT\_C <= '1';

end if;

else

CNT\_C <= '1';

CNT\_A <= "000";

end if;

end if;

end process;

CNT\_O <= std\_logic\_vector (CNT\_A);

end rtl;

## M2

Šis skaitiklis turi skaičiuoti moduliu 38. Jam reikės 6 bitų skaitiklio, kadangi 25 = 32 ir tai yra per mažai norint pavaizduoti skaičių 38. O 26 = 64, taigi reikės 6 bitų.

Skaitiklio M2 VHDL failas:

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.numeric\_std.all;

entity M2 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 M2;

architecture rtl of M2 is

signal CNT\_A : unsigned (5 downto 0);

begin

process (CLK, RST, CNT\_CMD)

begin

if RST = '1' then

CNT\_A <= "000000";

CNT\_C <= '1';

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

if CNT\_A < 37 then -- Pernasos signalas

CNT\_A <= CNT\_A + 1;

if CNT\_A = 36 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;

## M3

Šis skaitiklis turi skaičiuoti moduliu 60. Jau anksčiau parodžiau, kad maksimali 6 bitų skaičiaus vertė yra 64, o 5 bitų – 32. Taigi ir M3 skaitikliui reikės 6 bitų.

Skaitiklio M3 VHDL failas:

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.numeric\_std.all;

entity M3 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 M3;

architecture rtl of M3 is

signal CNT\_A : unsigned (5 downto 0);

begin

process (CLK, RST, CNT\_CMD)

begin

if RST = '1' then

CNT\_A <= "000000";

CNT\_C <= '1';

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

if CNT\_A < 59 then -- Pernasos signalas

CNT\_A <= CNT\_A + 1;

if CNT\_A = 58 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;

## JM2

Tai yra skaitiklis, kuriame naudojame visus tris sukurtus skaitiklius. Pagal užduotį, jis turėtų skaičiuoti iki 4560. Apskaičiuokime, ties kuriomis vertėmis turi būti M1, M2 ir M3 skaitikliai, kad JM2 skaičiuotų iki 4560 (*div* – dalyba paimant sveikąją dalį).

Taigi, galutinė M1 reikšmė – 0 (0002), M2 – 30 (0111102), M3 – 15 (0011112).

JM2 aprašymas VHDL kalba:

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.numeric\_std.all;

entity TOP\_CNT is port (

CLK\_TOP : in std\_logic; -- Sinchro signalas

RST\_TOP : in std\_logic; -- Reset signalas

Enable\_TOP : in std\_logic; -- Aktyvavimo signalas

Pernasa\_TOP : out std\_logic); -- Pernasa

end TOP\_CNT;

architecture struct of TOP\_CNT is

signal C, RST\_internal, C1, C2, C3 : std\_logic;

signal CNT\_1\_O : std\_logic\_vector (2 downto 0);

signal CNT\_2\_O : std\_logic\_vector (5 downto 0);

signal CNT\_3\_O : std\_logic\_vector (5 downto 0);

component M1 port (

CLK : in std\_logic; -- Sinchro signalas

RST : in std\_logic; -- Reset signalas

CNT\_CMD : in std\_logic; -- Aktyvavimo signalas

CNT\_C : out std\_logic; -- Pernasa

CNT\_O : out std\_logic\_vector (2 downto 0));

end component;

component M2 port (

CLK : in std\_logic; -- Sinchro signalas

RST : in std\_logic; -- Reset signalas

CNT\_CMD : in std\_logic; -- Aktyvavimo signalas

CNT\_C : out std\_logic; -- Pernasa

CNT\_O : out std\_logic\_vector (5 downto 0));

end component;

component M3 port (

CLK : in std\_logic; -- Sinchro signalas

RST : in std\_logic; -- Reset signalas

CNT\_CMD : in std\_logic; -- Aktyvavimo signalas

CNT\_C : out std\_logic; -- Pernasa

CNT\_O : out std\_logic\_vector (5 downto 0));

end component;

begin

CNT\_1 : M1 port map (CLK => CLK\_TOP,

RST => RST\_internal, CNT\_CMD => Enable\_TOP,

CNT\_C => C1, CNT\_O => CNT\_1\_O);

CNT\_2 : M2 port map (CLK => C1,

RST => RST\_internal, CNT\_CMD => Enable\_TOP,

CNT\_C => C2, CNT\_O => CNT\_2\_O);

CNT\_3 : M3 port map (CLK => C2,

RST => RST\_internal, CNT\_CMD => Enable\_TOP,

CNT\_C => C3, CNT\_O => CNT\_3\_O);

process (CLK\_TOP , RST\_TOP)

begin

if (RST\_TOP = '1') then

RST\_internal <= '1';

elsif CLK\_TOP'event and CLK\_TOP = '1' then

-- Stabdymas pasiekus JM2 verte

if ((CNT\_1\_O = "000")

and (CNT\_2\_O = "011110")

and (CNT\_3\_O = "001111")) then

RST\_internal <= '1';

Pernasa\_TOP <= '1';

else

RST\_internal <= '0';

Pernasa\_TOP <= '0';

end if;

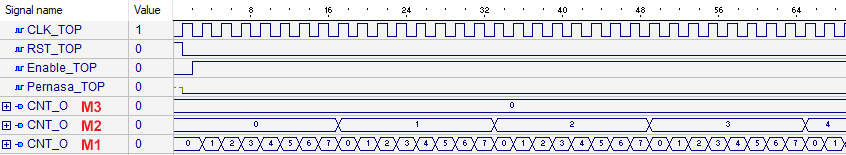
end if;

end process;

end struct;

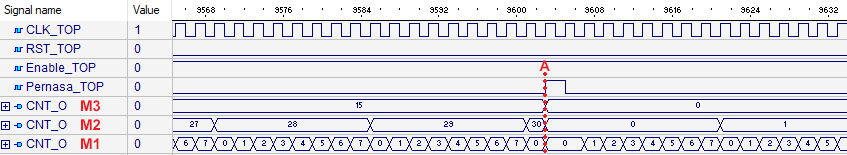
# Rezultatai

Pav. 1 matome laiko diagramos pradžią. Kai RST ir Enable signalai yra atitinkamai 0 ir 1, tuomet pradeda veikti skaitiklis. M1 reaguoja į kiekvieną kylantį CLK frontą; M2 padidėja vienetu, kai M1 pasiekia 7; ir, nors čia nesimato, M3 padidėja vienetu, kai M2 pasiekia 38.



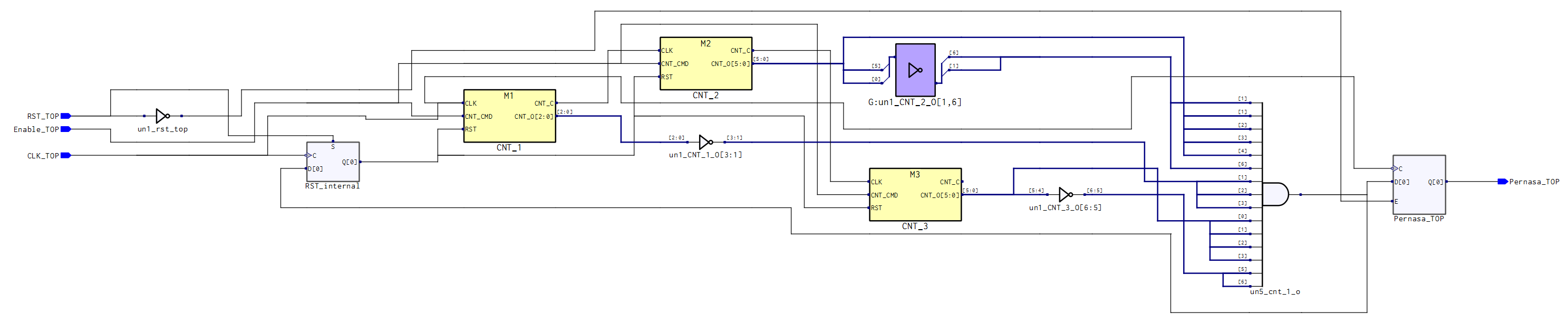
Pav. 1. Laiko diagramos pradžia.

Laiko diagramoje (Pav. 2) raudonai pažymėjau tašką A, ties kuriu M1 yra 0, M2 yra 30 ir M3 yra 15 (skaitiklis pasiekia 4560) ir sugeneruoja pernašos signalą kartu grįždamas į pradinę būseną.



Pav. 2. Laiko diagrama, kai skaitiklis pasiekia JM2.

Paleidus *„Simplify Pro“* įrankį galime pamatyti, kaip atrodo iš VHDL kodo sugeneruota schema.



Pav. 3. Schema, gauta iš „Simplify Pro“ įrankio.

# Perėjimas prie FPGA

Į FPGA plokštę kelsiu M1 skaitiklį, nes jo modulis mažiausias ir jį bus lengviausia pavaizduoti.

Tam, kad M1 skaitiklį galėtume įkelti į FPGA plokštę, jo kodą reikės truputį pakoreguoti. Pirmiausia pakeičiau RST signalo sąlygą. Po to reikėjo invertuoti skaitiklio išvesties vektorių, nes FPGA plokštėje LED užsidega, kai signalas būna 0. Taip atrodo pakoreguotas kodas:

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.numeric\_std.all;

entity M1 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 (2 downto 0));

end M1;

architecture rtl of M1 is

signal CNT\_A : unsigned (2 downto 0);

begin

process (CLK, RST, CNT\_CMD)

begin

if RST = '0' then

CNT\_A <= "000";

CNT\_C <= '1';

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

if CNT\_A < 7 then -- Pernasos signalas

CNT\_A <= CNT\_A + 1;

if CNT\_A = 6 then

CNT\_C <= '0';

else

CNT\_C <= '1';

end if;

else

CNT\_C <= '1';

CNT\_A <= "000";

end if;

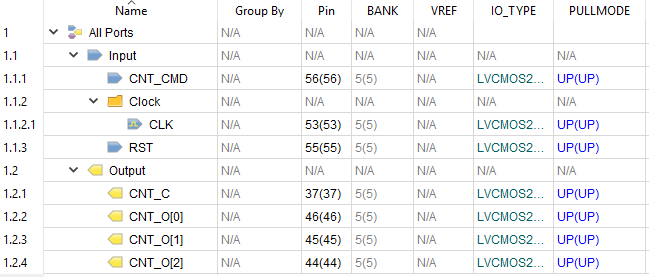
end if;

end process;

CNT\_O <= not(std\_logic\_vector (CNT\_A));

end rtl;

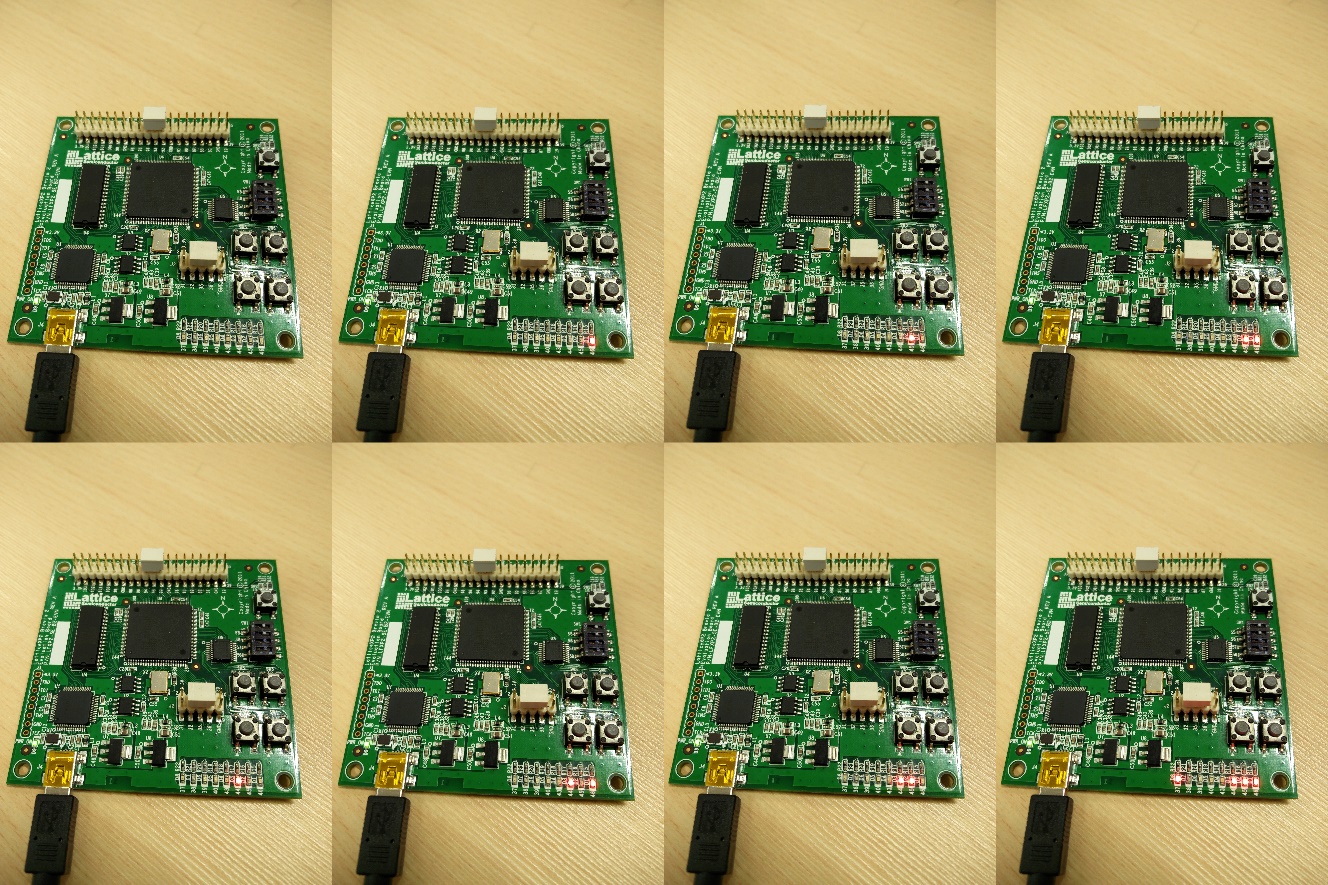
Po to priskiriame skaitiklio įvestis ir išvestis konkretiems fiziniams kontaktams per „Spreadsheet View“ įrankį.



Pav. 4. „Spreadsheet View“ langas.

Belieka sugeneruoti kodą ir užprogramuoti FPGA plokštę. Pateiksiu nuotraukas, kuriose matosi plokštės veikimas.

Jungiklyje uždėjau RST ir Enable signalus, kad nereikėtų laikyti nuspaustų mygtukų. Matome, kad kiekvieną kartą paspaudus CLK mygtuką, LED vaizduojamas dvejetainis kodas padidėja vienetu. Paskutinėje nuotraukoje pasiekiama vertė 7 ir sugeneruojamas pernašos signalas (pats kairiausias LED).



Pav. 5. Skaitiklio M1 veikimas FPGA plokštėje.

# Išvados

Darbas buvo atliktas sėkmingai; testuojant skaitiklį jis skaičiavo iki reikiamos reikšmės.

Tokią skaitiklio schemą nubraižyti ranka būtų ganėtinai sunku, tačiau pasinaudojus VHDL kalba, šią užduotį galima labai lengvai ir greitai atlikti. Tereikia pakoreguoti duotą kodą. Šis procesas ženkliai pagreitina schemų projektavimo laiką.