# KAUNO TECHNOLOGIJOS UNIVERSITETAS

## INFORMATIKOS FAKULTETAS

### KOMPIUTERIŲ KATEDRA


# SCHEMŲ APRAŠYMAS VHDL KALBA

4 Laboratorinis darbas


**Atliko:**

IFF 6/8 grupės stud.

Tadas Laurinaitis

**Priėmė**

Jaun. m. d. Lukas Romas



**KAUNAS, 2017**

# 1. ĮVADAS

## 1.1. DARBO TIKSLAS

Susipažinti su aparatūros aprašymo kalbomis ir jų galimybėmis. Aprašyti kombinacinę ir nuoseklios logikos schemą VHDL kalba, atlikti schemos sintezę ir išanalizuoti sintezės rezultatus.

## 1.2. UŽDUOTYS

1. VHDL kalba aprašyti kombinacinę schemą, kuri atlieka 1 laboratoriniame darbe nurodytą funkciją, ją ištestuoti, peržiūrėti sugeneruotą schemą.;

| | |
|---|---|
| 397 | 1,2,5,9,10,13,17,18,19,21,22,25,26,29,51,53,59,61 |

**1.1 pav. 1 laboratorinio darbo užduotis, kurioje nurodomi skaičiai, su kuriais kombinacinės schemos išvestis turi būti '1'.**

2. VHDL kalba aprašyti specializuotą postūmio registrą, kuris atlieka 3 darbo užduotyje nurodytus postūmius. Registrą ištestuoti ir peržiūrėti sugeneruotą schemą;

| | | | | | |
|---|---|---|---|---|---|
| 397 | 7 | LL1, CL1, AR2 | 0 | Sinchroninis | Papildomas |

**1.2 pav. Specializuoto registro aprašas, kuriame nurodomas skilčių skaičius (7), postūmio mikrooperacijos (LL1, CL1, AR2), įrašoma informacija (0), nulio nustatymas (sinchroninis) bei naudojamas skaičių kodas (papildomas) (2 schema).**

3. VHDL kalba aprašyti skaitiklio su lygiagrečiu užkrovimu schemą, skaitiklio modulis nurodytas 4 darbo užduotyje. Skaitiklį ištestuoti ir peržiūrėti sugeneruotą schemą;

| | |
|---|---|
| 397 | 36 |

**1.3 pav. 4 laboratorinio darbo užduotis, kurioje nurodytas skaitliuko modulis M (šiuo atveju 36).**

4. Naudojantis 3 punkte sukurta skaitiklio schema, sudaryti loginės PLIS matricos programą, ją išbandyti matricoje;

5. Parengti laboratorinio darbo ataskaitą. Joje pateikti sukurtų schemų aprašus, kompiliatoriaus panaudotų loginių elementų sąrašą ir laikines diagramas. Pateikti PLIS matricai pritaikytą schemą, aprašyti, kaip veikia stendas.

## 2. SCHEMŲ APRAŠYMAI VHDL KALBA

```
library IEEE ;
use IEEE . STD_LOGIC_1164 .ALL;

entity decoder is
port
(
    a , b , c , d , e , f : in std_logic ;
    y : out std_logic
);
end decoder ;

architecture beh of decoder is
signal xBus : std_logic_vector (5 downto 0);

begin
xBus <= a & b & c & d & e & f ;

with xBus select y <= '1' when "000001" ,
                      '1' when "000010" ,
                      '1' when "000101" ,
                      '1' when "001001" ,
                      '1' when "001010" ,
                      '1' when "001101" ,
                      '1' when "010001" ,
                      '1' when "010010" ,
                      '1' when "010011" ,
                      '1' when "010101" ,
                      '1' when "010110" ,
                      '1' when "011001" ,
                      '1' when "011010" ,
                      '1' when "011101" ,
                      '1' when "110011" ,
                      '1' when "110101" ,
                      '1' when "111011" ,
                      '1' when "111101" ,
                      '0' when others ;
end beh ;
```

**2.1 pav. Aprašyta pirmoji (kombinacinė) schema atliekanti pirmoje užduotyje nurodytą funkciją. Šiuo atveju a b c d e f atitinka x1 x2 x3 x4 x5 x6.**

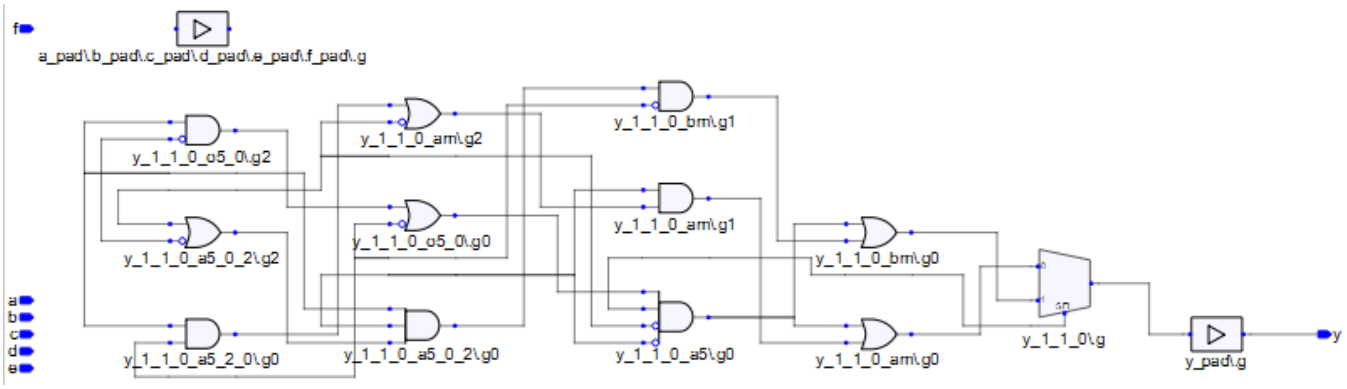
```
library IEEE ;
use IEEE . STD_LOGIC_1164 .ALL;

entity spec_register is
port
(
    clk , rst : in std_logic ;
    A0 , A1 : in std_logic ;
    Data : in std_logic_vector (6 downto 0);
    Q : out std_logic_vector (6 downto 0)
);
end spec_register ;

architecture beh2 of spec_register is
signal Qreg : std_logic_vector (6 downto 0);

begin
Q <= Qreg ;

process ( clk , rst )
begin

        if rst = '0' then                       --reset
            Qreg <= "0000000" ;
        elsif clk ' event and clk = '1' then
            if A0 = '0' and A1 = '0' then       --iraso
                Qreg <= Data ;
            elsif A0 = '0' and A1 = '1' then    --LL1
                Qreg <= Qreg (5 downto 0) & "0" ;
            elsif A0 = '1' and A1 = '0' then    --CL1
                Qreg <= Qreg (5 downto 0) & Qreg (6);
            elsif A0 = '1' and A1 = '1' then    --AR2
                Qreg <= Qreg (6) & Qreg (6)  & Qreg (6 downto 2) ;
        end if;
    end if;
end process ;
end beh2 ;
```

**2.2 pav. Aprašyta antroji schema (Specializuoto registro) atliekanti trečioje užduotyje nurodytus postūmius bei informacijos įrašymą.**

```
library ieee ;
use ieee . std_logic_1164 .all;
use ieee . std_logic_unsigned .all;
-------------------------------------------------
entity counter is

port
(
    clock : in std_logic ;
    clear : in std_logic ;
    load : in std_logic ;
    Data : in std_logic_vector (5 downto 0);
    p : out std_logic ;
    Q : out std_logic_vector (5 downto 0)
);
end counter ;

-- -------------------------------------------------

architecture behv of counter is

    signal Qreg : std_logic_vector (5 downto 0);

begin

    Q <= Qreg ;
    p <= '1' when Qreg = "100100" else '0';

    -- skaitiklio elgsenos aprasas

        process ( clock , clear )
        begin
            if clear = '0' then
                Qreg <= "000000" ;
            elsif ( clock ='1' and clock ' event ) then
                if Qreg = "100100" then
                    Qreg <= "000000" ;
                elsif ( load = '1') then
                    Qreg <= Data ;
                else
                    Qreg <= Qreg + 1;
            end if;
        end if;
    end process ;

end behv ;
```

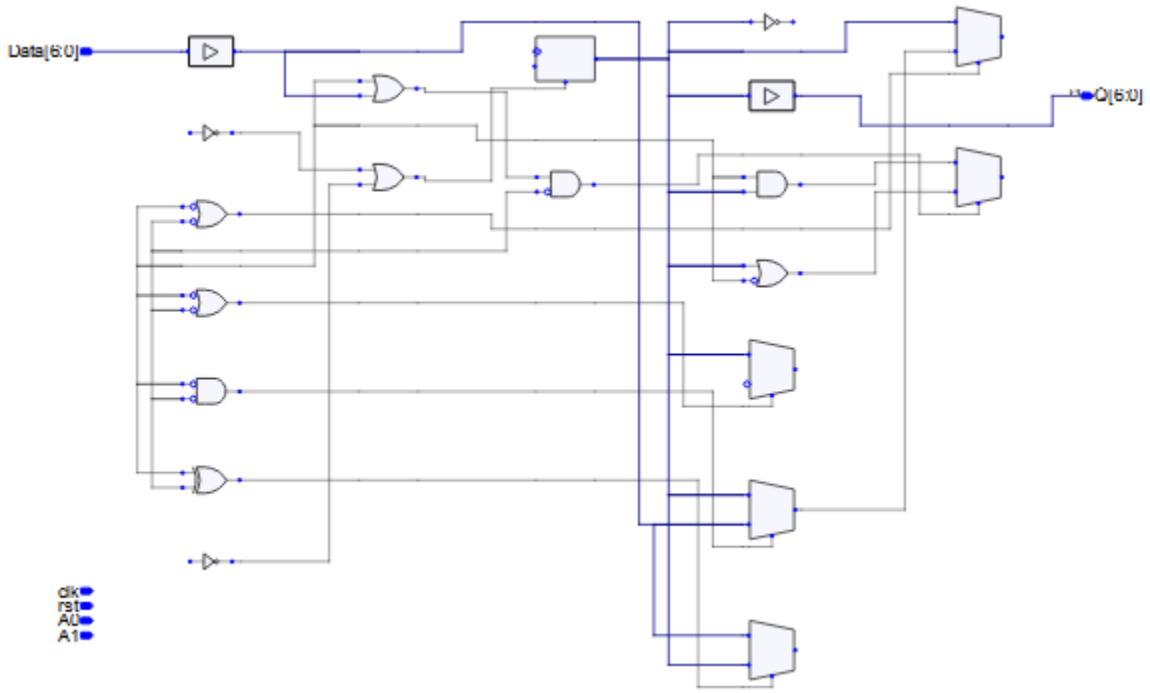
**2.3 pav. Aprašyta trečioji schema (skaitiklis) padaryta pagal ketvirtą užduotį.**

# 3. SUGENERUOTOS SCHEMOS



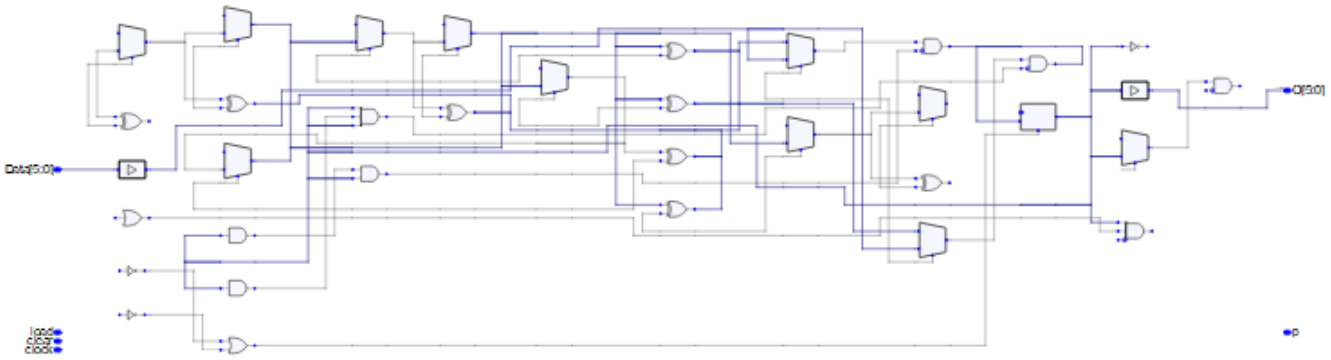

**3.1 pav. Pirmoji schema, sugeneruota pagal parašytą VHDL kodą vadovaujantis pirmos laboratorinio darbo užduoties sąlygomis.**

Pirmoji sugeneruota schema (3.1 pav.) palyginus su pirmo laboratorinio darbo metu daryta schema neatrodo sunkiau suprantama. Taip yra dėl to kad visos jungtys yra beveik aiškios bei naudojami nekomplikuoti elementai.




**3.2 pav. Antra schema, sugeneruota pagal parašytą VHDL kodą vadovaujantis trečios laboratorinio darbo užduoties sąlygomis.**

Antroji sugeneruota schema (3.2 pav.) palyginus su trečio laboratorinio metu daryta schema atrodo žymiai komplikuotesnė. Ši schema yra gerokai sunkiau žmogui suprantama dėl sudėtingų jungčių bei įvairių skirting elementų.



**3.3 pav. Trečioji schema (skaitiklis), sugeneruota pagal VHDL kodą vadovaujantis ketvirtos laboratorinio darbo užduoties sąlyga.**

Trečioji sugeneruota schema (3.3 pav) atrodo absoliučiai sudėtingai. Taip yra dėl daugybės sudėtingų elementų sujungtų sudėtingais būdais. Žmogui būtų labia sūdėtinga suprasti kas vyksta tokioje schemoje, tam prireiktų ne mažai laiko.

# 4. TESTAI IR REZULTATAI

```
testavimas : process
begin
    a <= '0'; b <= '0'; c <= '0'; d <= '0'; e <= '0'; f <= '1';     -- skaiciu su
    wait for 10 ns ;                                                -- kuriais y = 1
    a <= '0'; b <= '0'; c <= '0'; d <= '0'; e <= '1'; f <= '0';     -- pradzia.
    wait for 10 ns ;
    a <= '0'; b <= '0'; c <= '0'; d <= '1'; e <= '0'; f <= '1';
    wait for 10 ns ;
    a <= '0'; b <= '0'; c <= '1'; d <= '0'; e <= '0'; f <= '1';
    wait for 10 ns ;
    a <= '0'; b <= '0'; c <= '1'; d <= '0'; e <= '1'; f <= '0';
    wait for 10 ns ;
    a <= '0'; b <= '0'; c <= '1'; d <= '1'; e <= '0'; f <= '1';
    wait for 10 ns ;
    a <= '0'; b <= '1'; c <= '0'; d <= '0'; e <= '0'; f <= '1';
    wait for 10 ns ;
    a <= '0'; b <= '1'; c <= '0'; d <= '0'; e <= '1'; f <= '0';
    wait for 10 ns ;
    a <= '0'; b <= '1'; c <= '0'; d <= '0'; e <= '1'; f <= '1';
    wait for 10 ns ;
    a <= '0'; b <= '1'; c <= '0'; d <= '1'; e <= '0'; f <= '1';
    wait for 10 ns ;
    a <= '0'; b <= '1'; c <= '0'; d <= '1'; e <= '1'; f <= '0';
    wait for 10 ns ;
    a <= '0'; b <= '1'; c <= '1'; d <= '0'; e <= '0'; f <= '1';
    wait for 10 ns ;
    a <= '0'; b <= '1'; c <= '1'; d <= '0'; e <= '1'; f <= '0';
    wait for 10 ns ;
    a <= '0'; b <= '1'; c <= '1'; d <= '1'; e <= '0'; f <= '1';
    wait for 10 ns ;
    a <= '1'; b <= '1'; c <= '0'; d <= '0'; e <= '1'; f <= '1';
    wait for 10 ns ;
    a <= '1'; b <= '1'; c <= '0'; d <= '1'; e <= '0'; f <= '1';
    wait for 10 ns ;
    a <= '1'; b <= '1'; c <= '1'; d <= '0'; e <= '1'; f <= '1';
    wait for 10 ns ;
    a <= '1'; b <= '1'; c <= '1'; d <= '1'; e <= '0'; f <= '1';   -- skaiciai su
    wait for 10 ns ;                                               -- kuriais y = 1
                                                                   -- pabaiga
```

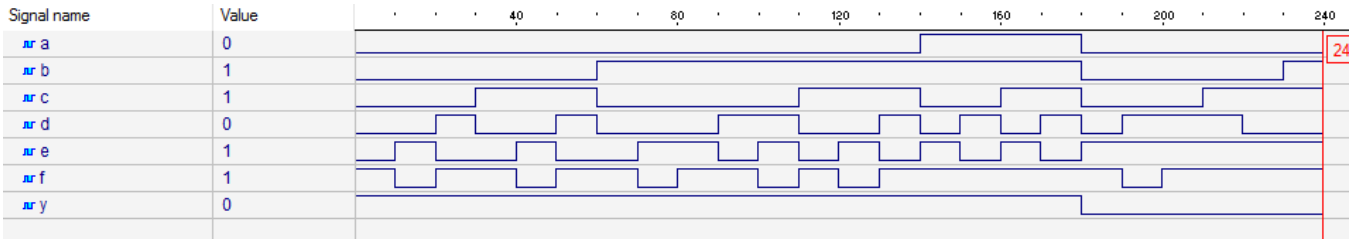
**4.1 pav. Pirmajai schemai testuoti aprašyti signalai**

```
ResetProcesas: process begin
rst <= '0';
wait for 3 ns; -- Reset signalas
rst <= '1';
wait;
end process;

Sinchroninis: process begin
clk <= '0';
wait for 5 ns;
clk <= '1';
wait for 5 ns;
end process;

Postumiai: process begin
Data <= "1101011";
-------------------------------------------------------
A0 <= '0'; A1 <= '0'; wait for 10 ns;        -- saugome
A0 <= '0'; A1 <= '1'; wait for 10 ns;        -- LL1
A0 <= '0'; A1 <= '0'; wait for 10 ns;        -- saugome
A0 <= '1'; A1 <= '0'; wait for 10 ns;        -- CL1
A0 <= '0'; A1 <= '0'; wait for 10 ns;        -- saugome
A0 <= '1'; A1 <= '1'; wait for 10 ns;        -- AR2
A0 <= '0'; A1 <= '0'; wait for 10 ns;        -- saugome
wait;
-- assert false report "Pabaiga" severity failure;
 end process;
```

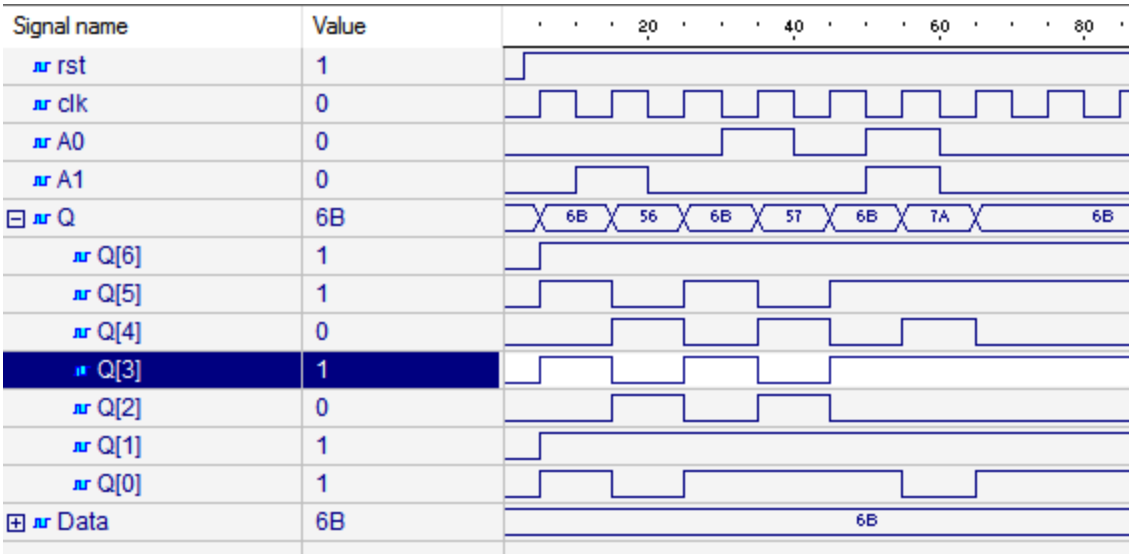
**4.2 pav. Antrajai schemai testuoti aprašyti signalai**

```
clock_procesas : process begin
    clock <= '0';
    wait for 8 ns;
    clock <= '1';
    wait for 8 ns;
end process;

Data_procesas : process begin
    Data <= "000001";
    wait;
end process;

clear_procesas : process begin
    clear <= '0';
    wait for 16 ns;
    clear <= '1';
    wait;
end process;

load_procesas : process begin
    load <= '0';
    wait for 100 ns;
    load <= '1';
    wait for 18 ns;
    load <= '0';
    wait;
end process;
```

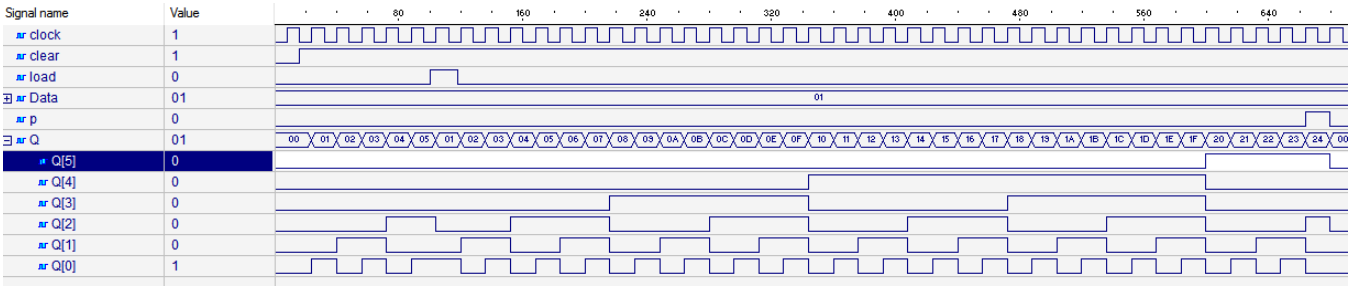
**4.3 pav. Trečiai schemai testuoti aprašyti signalai**




**4.4 pav. Pirmos schemos testavimo rezultatai. Nuo 0 ns iki 180 ns eina užduotyje nurodyti skaičiai, o nuo 180ns iki 240ns atsitiktiniai skaičiai ne lygus užduotyje nurodytiems.**




**4.5 pav. Antros schemos testavimo rezultatai. Iš pradžių nustatomas nulis reset (rst) pagalba, tada išsaugoma Data esanti informacija. Po to eina LL1 postūmis, reikšmių gražinimas į Data, CL1 postūmis, gražinimas, AR2 postūmis, gražinimas.**




**4.6 pav. Trečios schemos testavimo rezultatai.**

## 5. IŠVADOS

Buvo aprašytos ir ištestuotos 3 skirtingos schemos. Pirma schema veikė taip pat kaip ir pirmo laboratorinio darbo metu, sugeneravus nesiskyrė sunkumu, o antra schema veikė taip pat kaip ir trečio laboratorinio darbo metu, tačiau sugenereavus buvo žymiai sunkiau suprantama žmogui. Trečia schema buvo skaitiklis, kuris ištestuotas irgi veikė teisingai.

## LITERATŪRA


[1] Meng X. Implantable Wireless Devices for the Monitoring of Intracranial Pressure // X. Meng, U. Kawoos, S. M. Huang, M. R. Tofighi // IEEE 16th International Symposium. – 2012.

[2] North B. Intracranial pressure monitoring //P. Reilly, R. Bullock. Head Injury; Chapman & Hall, London, 1997.

[3] Popovic D. Noninvasive Monitoring of Intracranial Pressure / D. Popovic, M. Khoo, S. Lee // Recent Patents on Biomedical Engineering. – 2009, 2, p. 165-179.

[4] http://streetinfo.com/6-tips-hosting-great-open-house/