TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond
Arvutitehnika instituut

Merli Lall 142392 IAPB
Ragnar Sinikas 130736 IASB

NUTIKAS SÜSTEEM KOHIVAUTOMAAT

Praktikumi ülesande aruanne

Juhendaja: Priit Ruberg
Doktorant

Tallinn 2016

Autorideklaratsioon

Kinnitame, et oleme koostanud antud lõputöö iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.

Autorid: Merli Lall, Ragnar Sinikas

[21.11.2016]

Tallinn 2016

Annotatsioon

Praktikumitöö eesmärgiks oli luua nutikat süsteemi kirjeldav tõeväärtustabel, mis tuli minimeerida. Antud töös kasutati minimeerimiseks vastavat arvutiprogrammi Espresso. Minimeeritud lahend tuli kirjutada Hardware Description Language (VHDL) programmeerimiskeeles kasutades komponentne, mis olid kirjeldatud kasutades andmevoo, käitmuslikku ja struktuurset kirjeldusstiile. Komponentide hõlmamiseks tuli luua toplevel moodul. Kõikide komponentide samaaegseks kontrollimiseks tuli kirjutada testpink. Simulatsiooni kasutades sai veenduda töö tulemuse korrektsuses.

Töö on kirjutatud eesti keeles ning sisaldab teksti 9 leheküljel, 9 peatükki, 2 pilti, 1 tabelit.

Sisukord

Sissejuhatus

Ülesandeks on nutikas süsteem, antud töös on selleks kohvimasina jookide valikusüsteemile vastavate väljundite genereerimine, millele tuleb luua minimaalne juhtloogika VHDL keeles. Ülesandes on antud sisendid ning väljundid, kuid väljundfunktsioon tuleb meeskondadel endal koostada lähtudes ülesande tingimustest

Lähteülesanne

Kohivautomaat saab informatsiooni 3 andurilt: veetaseme andur, vee temperatuuriandur, joogitüübi valik erkaanilt. Veetaseme anduril on 2 olekut: madal, kõrge. Vee temperatuurianduril on 2 olekut: madal, kõrge. Joogitüübil saab olla 3 võimalust: kuum vesi, must kohv, piimaga kohv. Kohviautomaadi väljunditeks on küttekeha, veekraan, piimakraan. Küttekeha saab olla kas sisse lülitatud või välja lülitatud. Veekraan saab olla kas kinni või lahti. Piimakraan saab olla kas kinni või lahti. Vastavalt ettenatud tingimustele sünteesida kohviautomaadi juhtimissüsteem.

Põhjendatud väljundfunktsioonid

.SISENDID

x1 -veetaseme andur -> 0 või 1 -> madal või kõrge

x2 -vee temp andur -> 0 või 1 -> madal või kõrge

x3 -joogitüübi kõrgem bit

x4 -joogitüübi madalam bit

JOOGITÜÜBID

00 - kuum vesi

01 - must kohv

10 – piimakohv

VÄLJUNDID

y1 – küttekeha -> 0 või 1 -> madal või kõrge

y2 – veekraan -> 0 või 1 -> madal või kõrge

y3 – piimakraan -> 0 või 1 -> madal või kõrge

Kõigil kolmel väljundil on kaks olekut 0 või 1ehk, kas kraan kinni või lahti.
Lisaks peab alati veetaseme andur x1 olema kõrge (1), muidu masin tööd teha ei saa, sest vett pole aparaadis.

Kui joogi tüübiks on valitud kuum vesi, mille kood on 00 siis väljundid peavad olema

y1 = 1 // meil on vaja vesi kuumaks ajada

y2 = 1 // veekraan tehakse lahti

y3 = 0 // kuna kasutaja soovis kuuma vett siis piima talle ei anna

Kui joogi tüübiks on valitud must kohv, mille kood on 01 siis väljundid peavad olema

y1 = 1 // kohvi jaoks on vaja soovitatavalt kuuma vett

y2 = 1 // veekraan tehakse lahti

y3 = 0 // kuna kasutaja soovis musta kohvi siis piima kraan jääb kinni

NB! kohvi kohta polnud andmeid ja selle tõttu ei kajastu see siin ka

Kui joogi tüübiks on valitud piimakohv, mille kood on 10 siis väljundid peavad olema

y1 = 1 // küttekeha sisse, et kuuma vett saada

y2 = 1 // veekraan lahti

y3 = 1 // kuna kasutaja soovis piimakohvi siis piima kraan peab olema ka avatud

---

Väljundfunktsioonide tabel

x1x2x3x4	y1y2y3
0001	000
0010	000
0011	000
0100	000
0101	000
0110	000
0111	000

Kõik need juhud kus x1 on 0 ehk veetaseme andur on madal on ka väljundid 0, kuna kuuma vett ega kohvi ei saa valmistada.

1000	110
1001	110
1010	111
1011	000
1100	010
1101	010
1110	011

1111 000 // x3x4 on 11, selline joogi valik puudub, sellest tingitult väljundid 0

Minimaalne espresso tulemus

Vaikusse jäid phase 000 ja 001.Valisime phase 000, sest sellel oli väikseim total ja lihtsam arvestada, et kõik väljundfunktsioonid on inventeeritud.

Espresso tulemus väljakirjutatult (x4 – noorim bit):

y1i = x3x4 + x2 + x1i

y2i = x3x4 + x1i

y3i = x3x4 + x1i + x3i

```
# phase is ---- 000
# ESPRESSO      Time was 0.00 sec, cost is c=4(0) in=5 out=8 tot=13
# phase is ---- 001
# ESPRESSO      Time was 0.00 sec, cost is c=4(0) in=7 out=6 tot=13
# phase is ---- 010
# ESPRESSO      Time was 0.00 sec, cost is c=5(0) in=8 out=8 tot=16
# phase is ---- 011
# ESPRESSO      Time was 0.00 sec, cost is c=5(0) in=9 out=6 tot=15
# phase is ---- 100
# ESPRESSO      Time was 0.00 sec, cost is c=5(0) in=10 out=7 tot=17
# phase is ---- 101
# ESPRESSO      Time was 0.00 sec, cost is c=5(0) in=12 out=5 tot=17
# phase is ---- 110
# ESPRESSO      Time was 0.00 sec, cost is c=6(0) in=12 out=7 tot=19
# phase is ---- 111
# ESPRESSO      Time was 0.00 sec, cost is c=4(0) in=11 out=5 tot=16
# opoall        Time was 0.00 sec, cost is c=4(0) in=5 out=8 tot=13
.i 4
.o 3
#.phase 000
.p 4
--11 111
-1-- 100
--0- 001
0--- 111
.e
```

VHDL programmi olulised ja kommenteeritud lõigud

Alustame tagantpoolt, kõige olulisemaks on signaalide nimetused, mida antud koodis kasutasime, et oleks hiljem kõike lihtsam mõista.

Olulised lõigud test.vhd-s:

```
architecture bench of test_bench_system is
  --signaalidele nimetuste andmine, kuna oleme eelnevalt ära kasutanud
soodsamad nimetused, siis sisenditeks on n1..n4
  -- q1_tt, q2_tt, q3_tt - toevaartustabeli väljundsignaalid
  -- q1_esp, q2_esp, q3_esp - andmevoo kirjeldustiilis Espresso väljundid
  -- q1_beh, q2_beh, q3_beh - käitumuslikus kirjeldusstiilis Espresso
väljundid
  -- q1_str, q2_str, q3_str - struktuurses kirjeldusstiilis Espresso väljundid
  signal n1, n2, n3, n4, q1_tt, q2_tt, q3_tt, q1_esp, q2_esp, q3_esp, q1_beh,
q2_beh, q3_beh, q1_str, q2_str, q3_str: std_logic;

  -- test_systemi component kirjeldab ära sisendid ja väljundid
simuleerimiseks
  component test_system
  port ( i1, i2, i3, i4: in std_logic;
         o1, o2, o3, o1_esp, o2_esp, o3_esp, o1_beh, o2_beh, o3_beh, o1_str,
o2_str, o3_str : out std_logic );
  end component;
  --Kasutame topleveli entity-t
  for UN1: test_system use entity work.toplevel_system(toplevel);
begin
  -- - Sisendsignaalide genereerimine peale igat 10 ns
  n1 <= '0' after 0 ns, '1' after 80 ns, '0' after 160 ns;
  n2 <= '0' after 0 ns, '1' after 40 ns, '0' after 80 ns, '1' after 120 ns;
  n3 <= '0' after 0 ns, '1' after 20 ns, '0' after 40 ns, '1' after 60 ns,
        '0' after 80 ns, '1' after 100 ns, '0' after 120 ns, '1' after 140 ns;
  n4 <= '0' after 0 ns, '1' after 10 ns, '0' after 20 ns, '1' after 30 ns,
        '0' after 40 ns, '1' after 50 ns, '0' after 60 ns, '1' after 70 ns,
        '0' after 80 ns, '1' after 90 ns, '0' after 100 ns, '1' after 110 ns,
        '0' after 120 ns, '1' after 130 ns, '0' after 140 ns, '1' after 150 ns;

  -- Seome kirjeldatud sisendid ja väljundid pärissignaalidega (selle all
mõtlen genereeritud signaalid, millega testime ja väljundid mida hiljem saame
testpingis)
  UN1: test_system port map (n1, n2, n3, n4, q1_tt, q2_tt, q3_tt, q1_esp,
q2_esp, q3_esp, q1_beh, q2_beh, q3_beh, q1_str, q2_str, q3_str);
end architecture bench;
```

Olulised lõigud toplevel.vhd-s:

```
--Entity deklaratsioon: sisend ja väljundsignaalid
entity toplevel_system is
  port ( i1, i2, i3, i4: in std_logic;
         o1, o2, o3, o1_esp, o2_esp, o3_esp, o1_beh, o2_beh, o3_beh, o1_str, o2_str,
o3_str : out std_logic );
end entity toplevel_system;

[...]

--Loodud on iga alamkomponendi jaoks eraldi entity-d, mida välja kutsun.
  for U1: tab use entity work.tabel_system(tabel);
  for U2: esp use entity work.espresso_system(espresso);
  for U3: behav use entity work.behaviour_system(behavioural);
  for U4: struct use entity work.structural_system(structure);
```

```
begin
 --Port map - mapping of the components and entity ports (tegelikud sisendid,
valjundid skeemis)
  --Reaalsed sisendid ja väljundid komponentidele kasutades port map-i - igal
komponendil on 4 sisendit ja 3 väljundit
  U1: tab port map (i1, i2, i3, i4, o1, o2, o3);
  U2: esp port map (i1, i2, i3, i4, o1_esp, o2_esp, o3_esp);
  U3: behav port map (i1, i2, i3, i4, o1_beh, o2_beh, o3_beh);
  U4: struct port map (i1, i2, i3, i4, o1_str, o2_str, o3_str);
end architecture toplevel;
```

Olulised lõigud TT_tabel.vhd-s:

```
--tabeli entity defineerimine (kõik teised alamkomponendid on sarnase
ehitusega ja ei hakka teiste koodide puhul siin eraldi kordama):
entity tabel_system is
  port ( x1, x2, x3, x4: in std_logic;
         y1, y2, y3: out std_logic );
end entity tabel_system;

[...]

  --Oluline see, et sisendsignaali muutuja on 4 bitine ja valjundsignaali
muutuja on 3 bitine.
    variable in_word: std_logic_vector (3 downto 0);
    variable out_word: std_logic_vector(2 downto 0);
--Ülejäänud osa on väga sarnane näitekoodidega, mis meile anti ja võib lugeda
lõpust.
```

Olulised lõigud espresso.vhd-s:

```
--Väga sarnane kodutöö I ülesandele, kood jäljendas espressost saadud
vastuseid, ei oska midagi väga olulist ja uut siia välja tuua ja koodi võib
vaadata lisast.
```

Olulised lõigud behav.vhd-s:

```
architecture behavioural of behaviour_system is
    begin
        process (x1,x2,x3,x4)
        begin
        --phase 000, koik signaalid olgu esialgu uhed, kui midagi muutub, siis
muudetakse vastupidiseks
            y1_beh <= '1';
            y2_beh <= '1';
            y3_beh <= '1';

        if ((x3 = '1' and x4 = '1') or (x1 = '0')) then
            y1_beh <= '0';
            y2_beh <= '0';
            y3_beh <= '0';
        end if;
        if (x2 = '1') then
            y1_beh <= '0';
        end if;
        if (x3 = '0') then
            y3_beh <= '0';
        end if;
    end process;
end architecture behavioural;
```

Olulised lõigud struct.vhd-s:

```
--tuleb igale ventiilile valja kutsuda tema entity
for gate1: not1 use entity work.not1(normal);
for gate2: and2 use entity work.and2(normal);
for gate3: not1 use entity work.not1(normal);
for gate4: nor3 use entity work.nor3(normal);
for gate5: nor2 use entity work.nor2(normal);
for gate6: nor3 use entity work.nor3(normal);

--3)Sisemised ühendused
--Meil on 6 ventiili
--Port map - mapping of the components and entity ports (tegelikud sisendid,
valjundid skeemis)
begin
    gate1: not1 port map(x1, i);
    gate2: and2 port map(x3,x4, j);
    gate3: not1 port map(x3, k);
    gate4: nor3 port map(i, x2, j, y1_str);
    gate5: nor2 port map(j, i, y2_str);
    gate6: nor3 port map(j, k, i, y3_str);

--Ülejäänud koodi näeb lisades, seal on ka kommentaare.
```

Töös esinenud rasked kohad, probleemid

Antud töös oli kindlasti üks raskemaid kohti VHDL koodi kirjutamine. Esimesest kodutöös olid tuttavad testpink, andmevoo kirjeldusstiil ja tõeväärtustabeli koostamine. Käitumusliku stiili kirjutamisega sai hakkama ka suhteliselt kergesti, oli veidi harjumatu signaalide kirjeldamine. Struktuurse kirjeldusstiili kirjutamine nõudis aga põhjalikumat teadmist VHDL-i komponentide, entityde ja nendest tulenevate reeglite tundmist ning see võttis palju aega. Näiteid ei olnud piisavalt (Muxi näide oli liiga väike võrreldes sellega, mida ise pidime tegema) ning kuna meie praktikum on nädalas üks varasemaid, siis päev enne õppejõudude lisatud materjali (loengus tutvustatud lehekülg) ei olnud kasu, sest pidime informatsiooni ise leidma.

Kuigi testpingi kasutamist oleme varasemalt teinud, ei olnud meile hästi seletatud seda, kuidas mitut erinevat funktsiooni siduvat toplevelit saaks testpingis simuleerida, seega esialgu tekkisid probleemid ka seal, kuid saime nendest jagu. Segaseks jäi see, kas pidi testbenchis simuleerima funktsioone või pidi seda kirjeldama FPGA plaadi jaoks sisend-väljunditena, sest kuidas simulatsiooni ja FPGA plaadi koode kokku ühte faili kirjutada on endiselt teadmata. Antud juhul hetkel on meil ainult simuleerimiseks kirjutatud kood.

Kokkuvõte

Simulatsioon töötab vastavalt meie kirjutatud juhistele. Seega oleme oma tulemusega rahul. Õppisime palju VHDL-i kohta.

Kasutatud kirjandus

1. Green Mountain Computing Systems, Behavioral Descriptions
http://www.gmvhdl.com/textio.htm

2. Surf VHDL, Structural Modeling
http://surf-vhdl.com/vhdl-syntax-web-course-surf-vhdl/vhdl-structural-modeling-style/

3. YouTube Alireza Saberi, Structural VHDL
https://www.youtube.com/watch?v=Y8YMs74dyfM

4. YouTube engineeringstudy, entity declaration in vhdl
https://www.youtube.com/watch?annotation_id=annotation_3173389659&feature=iv&src_vid=YQVRnyrdQxY&v=x5Yk2azRSg8

5. “Riistvara kirjeldamiskeel VHDL” – K. Tammemäe, Tallinn 2002

Lisa – VHDL kood

```
--------------------------------------------------------------------------------
-- IAY0105 - Test bench
--------------------------------------------------------------------------------
--Merli Lall, Ragnar Sinikas 2016
--------------------------------------------------------------------------------
library IEEE; use IEEE.std_logic_1164.all;
--test benchi entity on ka varasemates oppejoudude poolt kirjutatud
naidetes niimoodi kirjeldatud:
entity test_bench_system is
end entity test_bench_system;

library IEEE; use IEEE.std_logic_1164.all;
architecture bench of test_bench_system is
  --signaalidele nimetuste andmine, kuna olen eelnevalt ara kasutanud
soodsamad nimetused, siis sisenditeks on n1..n4
  -- q1_tt, q2_tt, q3_tt - toevaartustabeli valjundsignaalid
  -- q1_esp, q2_esp, q3_esp - andmevoo kirjeldustiilis Espresso
valjundid
  -- q1_beh, q2_beh, q3_beh - kaitumuslikus kirjeldusstiilis Espresso
valjundid
  -- q1_str, q2_str, q3_str - struktuurses kirjeldusstiilis Espresso
valjundid
  signal n1, n2, n3, n4, q1_tt, q2_tt, q3_tt, q1_esp, q2_esp, q3_esp,
q1_beh, q2_beh, q3_beh, q1_str, q2_str, q3_str: std_logic;

  -- test_systemi component kirjeldab ara sisendid ja valjundid
simuleerimiseks
  component test_system
  port ( i1, i2, i3, i4: in std_logic;
         o1, o2, o3, o1_esp, o2_esp, o3_esp, o1_beh, o2_beh, o3_beh,
o1_str, o2_str, o3_str : out std_logic );
  end component;

  --Kasutame topleveli entityt
  for UN1: test_system use entity work.toplevel_system(toplevel);
begin
  -- Input signaals (after every 10 ns) - signaalide genereerimine
  n1 <= '0' after 0 ns, '1' after 80 ns, '0' after 160 ns;
  n2 <= '0' after 0 ns, '1' after 40 ns, '0' after 80 ns, '1' after
120 ns;
  n3 <= '0' after 0 ns, '1' after 20 ns, '0' after 40 ns, '1' after 60
ns,
        '0' after 80 ns, '1' after 100 ns, '0' after 120 ns, '1' after
140 ns;
  n4 <= '0' after 0 ns, '1' after 10 ns, '0' after 20 ns, '1' after 30
ns,
        '0' after 40 ns, '1' after 50 ns, '0' after 60 ns, '1' after 70
ns,
        '0' after 80 ns, '1' after 90 ns, '0' after 100 ns, '1' after
110 ns,
        '0' after 120 ns, '1' after 130 ns, '0' after 140 ns, '1' after
150 ns;

  -- Seome kirjeldatud sisendid ja valjundid parissignaalidega
  UN1: test_system port map (n1, n2, n3, n4, q1_tt, q2_tt, q3_tt,
q1_esp, q2_esp, q3_esp, q1_beh, q2_beh, q3_beh, q1_str, q2_str,
q3_str);
end architecture bench;
```

```
--------------------------------------------------------------------------
--
-- IAY0150 - Toplevel
--------------------------------------------------------------------------
--
-- Merli Lall, Ragnar Sinikas 2016
--------------------------------------------------------------------------
--
library IEEE; use IEEE.std_logic_1164.all;
--entity deklaratsioon: sisend ja valjundsignaalid
entity toplevel_system is
  port ( i1, i2, i3, i4: in std_logic;
         o1, o2, o3, o1_esp, o2_esp, o3_esp, o1_beh, o2_beh, o3_beh,
o1_str, o2_str, o3_str : out std_logic );
end entity toplevel_system;


architecture toplevel of toplevel_system is

  component tab
  port ( x1, x2, x3, x4: in std_logic;
         y1, y2, y3: out std_logic );
  end component;
  component dataflow
  port ( x1, x2, x3, x4: in std_logic;
         y1, y2, y3: out std_logic );
  end component;

  component behav
  port ( x1, x2, x3, x4: in std_logic;
         y1_beh, y2_beh, y3_beh: out std_logic );
  end component;

  component struct
  port ( x1, x2, x3, x4: in std_logic;
         y1_str, y2_str, y3_str: out std_logic );
  end component;

begin

  --Reaalsed sisendid ja valjundid komponentidele
  U1: tab port map (i1, i2, i3, i4, o1, o2, o3);
  U2: dataflow port map (i1, i2, i3, i4, o1_esp, o2_esp, o3_esp);
  U3: behav port map (i1, i2, i3, i4, o1_beh, o2_beh, o3_beh);
  U4: struct port map (i1, i2, i3, i4, o1_str, o2_str, o3_str);
end architecture toplevel;
```

```
--------------------------------------------------------------------------
--
-- IAY0150 - Truth table.
--------------------------------------------------------------------------
--
-- Merli Lall, Ragnar Sinikas 2016
--------------------------------------------------------------------------
--
library IEEE; use IEEE.std_logic_1164.all;
entity tab is
  port ( x1, x2, x3, x4: in std_logic;
         y1, y2, y3: out std_logic );
end entity tab;


architecture tabel of tab is
begin
  process (x1, x2, x3, x4)
  --sisendsignaali muutuja on 4 bitine ja valjundsignaali muutuja on 3
bitine.
    variable in_word: std_logic_vector (3 downto 0);
    variable out_word: std_logic_vector(2 downto 0);
  begin
    in_word := x1 & x2 & x3 & x4;
    case  in_word  is
      when "0000" => out_word := "000";
      when "0001" => out_word := "000";
      when "0010" => out_word := "000";
      when "0011" => out_word := "000";
      when "0100" => out_word := "000";
      when "0101" => out_word := "000";
      when "0110" => out_word := "000";
      when "0111" => out_word := "000";
      when "1000" => out_word := "110";
      when "1001" => out_word := "110";
      when "1010" => out_word := "111";
      when "1011" => out_word := "000";
      when "1100" => out_word := "010";
      when "1101" => out_word := "010";
      when "1110" => out_word := "011";
      when "1111" => out_word := "000";
      when others => out_word := "---";
    end case;
    y1 <= out_word(2);    y2 <= out_word(1);
    y3 <= out_word(0);
  end process;
end architecture tabel;
```

```
--------------------------------------------------------------------------
--
-- IAY0150 -  Espresso result : data flow
--------------------------------------------------------------------------
--
-- Merli Lall, Ragnar Sinikas 2016
--------------------------------------------------------------------------
--
library IEEE; use IEEE.std_logic_1164.all;
entity dataflow is --dataflow
  port ( x1, x2, x3, x4: in std_logic;
         y1, y2, y3: out std_logic );
end entity dataflow;

architecture espresso of dataflow is
begin
  y1 <= not((x3 and x4) or x2 or (not x1));
  y2 <= not((x3 and x4) or (not x1));
  y3 <= not((x3 and x4) or (not x3) or (not x1));

end architecture espresso;


--------------------------------------------------------------------------
--
-- IAY0150 -  Espresso result, behavioural model
--------------------------------------------------------------------------
--
-- Merli Lall, Ragnar Sinikas 2016
--------------------------------------------------------------------------
--
library IEEE; use IEEE.std_logic_1164.all;
entity behav is
  port ( x1, x2, x3, x4: in std_logic;
         y1_beh, y2_beh, y3_beh: out std_logic );
end entity behav;

architecture behavioural of behav is
    begin
        process (x1,x2,x3,x4)
        begin
        --phase 000, koik signaalid olgu esialgu uhed, kui midagi
muutub, siis muudetakse vastupidiseks
            y1_beh <= '1';
            y2_beh <= '1';
            y3_beh <= '1';

        if ((x3 = '1' and x4 = '1') or (x1 = '0')) then
            y1_beh <= '0';
            y2_beh <= '0';
            y3_beh <= '0';
        end if;
        if (x2 = '1') then
            y1_beh <= '0';
        end if;
        if (x3 = '0') then
            y3_beh <= '0';
        end if;
    end process;
end architecture behavioural;
```

```
--------------------------------------------------------------------------
--
-- IAY0150 - Espresso: Structural model
--------------------------------------------------------------------------
--
-- Merli Lall, Ragnar Sinikas 2016
--------------------------------------------------------------------------
--
library IEEE; use IEEE.std_logic_1164.all;
--entity deklaratsioon: sisend ja valjund signaalid
entity struct is
  port ( x1, x2, x3, x4: in std_logic;
         y1_str, y2_str, y3_str: out std_logic );
end entity struct;

--structural model:
--1)komponentide deklaratsioon (gates used in the circuit) - mida
kasutati: --NOT --AND --2-NOR --3-NOR
--Ei tohi omada sama sisend ja valjundnimetusi mis entity-l on.

architecture structure of struct is
component not1
port(
    a : in std_logic;
    b : out std_logic);
end component;
component and2
port(
    c, d : in std_logic;
    e : out std_logic);
end component;
component nor2
port(
    f, g : in std_logic;
    h : out std_logic);
end component;
component nor3
port(
    l, m, n : in std_logic;
    o : out std_logic);
end component;

--2)Signaalide deklaratsioon(intermediates of the circuit):
signal i, j, k: std_logic;

--3)Sisemised Ć¼hendused
--Meil on 6 ventiili
--Port map - mapping of the components and entity ports (tegelikud
sisendid, valjundid skeemis)
begin
    gate1: not1 port map(x1, i);
    gate2: and2 port map(x3,x4, j);
    gate3: not1 port map(x3, k);
    gate4: nor3 port map(i, x2, j, y1_str);
    gate5: nor2 port map(j, i, y2_str);
    gate6: nor3 port map(j, k, i, y3_str);


end structure;
```

```
 ----------------------------------------------------------------------
-
-- IAY0150 - Espresso: Structural model : not
----------------------------------------------------------------------
-- Merli Lall, Ragnar Sinikas 2016
----------------------------------------------------------------------
library IEEE; use IEEE.std_logic_1164.all;

entity not1 is
    port(a: in std_logic;
         b: out std_logic);
    end not1;

architecture normal of not1 is
begin
    b <= not a;
end normal;

----------------------------------------------------------------------
-- IAY0150 - Espresso: Structural model : 2-and
----------------------------------------------------------------------
-- Merli Lall, Ragnar Sinikas 2016
----------------------------------------------------------------------
library IEEE; use IEEE.std_logic_1164.all;

entity and2 is
    port(
        c : in std_logic;
        d : in std_logic;
        e : out std_logic);
    end and2;

architecture normal of and2 is
begin
    e <= c and d;
end normal;

----------------------------------------------------------------------
--  IAY0150 - Espresso: Structural model : 2-nor
----------------------------------------------------------------------
-- Merli Lall, Ragnar Sinikas 2016
----------------------------------------------------------------------
library IEEE; use IEEE.std_logic_1164.all;

entity nor2 is
    port(
        f : in std_logic;
        g : in std_logic;
        h : out std_logic);
    end nor2;

architecture normal of nor2 is
begin
    h <= not(f or g);
end normal;
```

```
--------------------------------------------------------------------------
--  IAY0150 - Espresso: Structural model : 3-nor
--------------------------------------------------------------------------
-- Merli Lall, Ragnar Sinikas 2016
--------------------------------------------------------------------------
library IEEE; use IEEE.std_logic_1164.all;
entity nor3 is
    port(
        l, m, n : in std_logic;
        o : out std_logic);
    end nor3;

architecture normal of nor3 is
begin
    o <= not( l or m or n);
end normal;
```

Pilt simulatsioonist: