



**ЕЛЕКТРОТЕХНИЧКИ ФАКУЛТЕТ
УНИВЕРЗИТЕТ У БАЊОЈ ЛУЦИ**

**ПРОЈЕКТОВАЊЕ ДИГИТАЛНИХ СИСТЕМА
ИЗВЈЕШТАЈ СА ЛАБОРАТОРИЈСКИХ
ВЈЕЖБИ**

Студент: Марко Грумић
Број индекса: 1120/11

Предметни наставник: др Бранко Докић
Предметни асистент: Жељко Ивановић

Број бодова:

Бања Лука
Септембар, 2016. године

Вјежба бр. 1

VHDL – Комбинациона мрежа

1. Кратак опис вјежбе

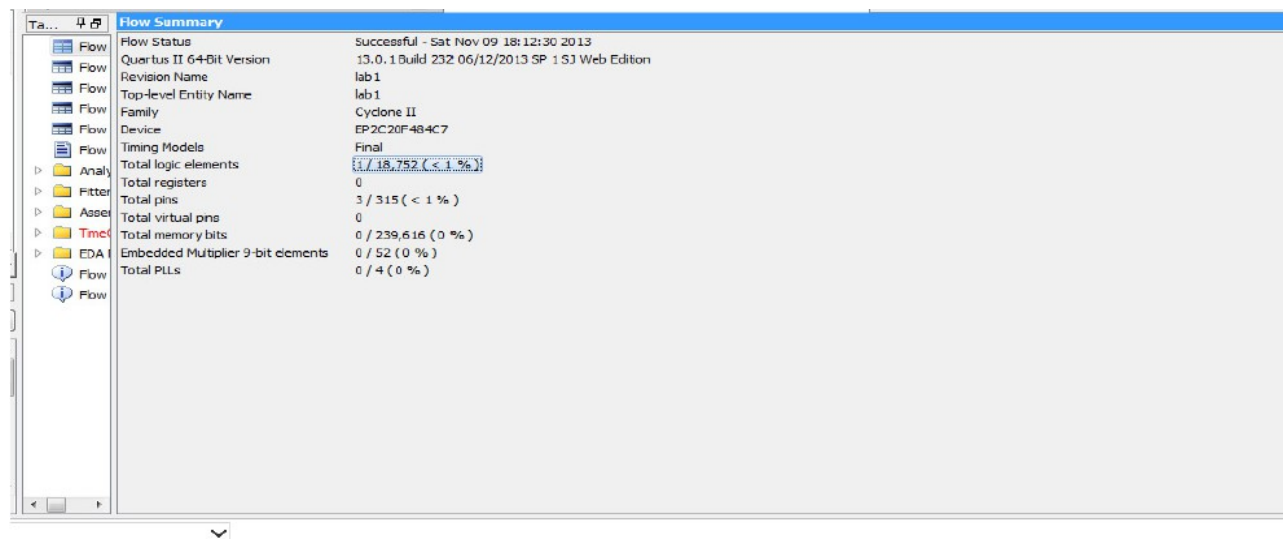
Циљ ове вјежбе је реализовање једноставне комбинационе мреже (2XOR), симулација добијеног дизајна и верификација употребом развојног окружења.

2. Кориштени софтвер и опрема

За реализацију је кориштен програмски пакет **Quartus II** произвођача Altera као и развојно окружење DE1 од истог произвођача.

3. Резултати и закључак

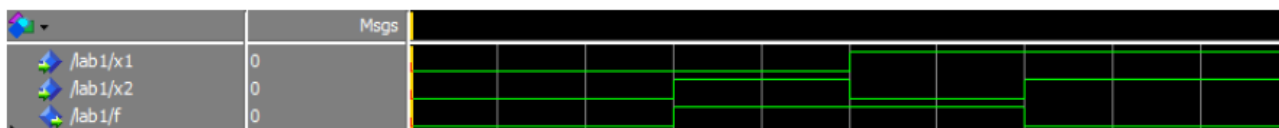
За реализацију потребан је један логички елемент што потврђује *Compilation Report*:



Flow Summary	
Flow Status	Successful - Sat Nov 09 18:12:30 2013
Quartus II 64-bit Version	13.0.1 Build 232 06/12/2013 SP 1 S3 Web Edition
Revision Name	lab1
Top-level Entity Name	lab1
Family	Cyclone II
Device	EP2K10K10-10
Timing Models	Final
Total logic elements	1 / 18,752 (< 1 %)
Total registers	0
Total pins	3 / 315 (< 1 %)
Total virtual pins	0
Total memory bits	0 / 239,616 (0 %)
Embedded Multiplier 9-bit elements	0 / 52 (0 %)
Total PLLs	0 / 4 (0 %)

Сл. 1.1 Flow Summary

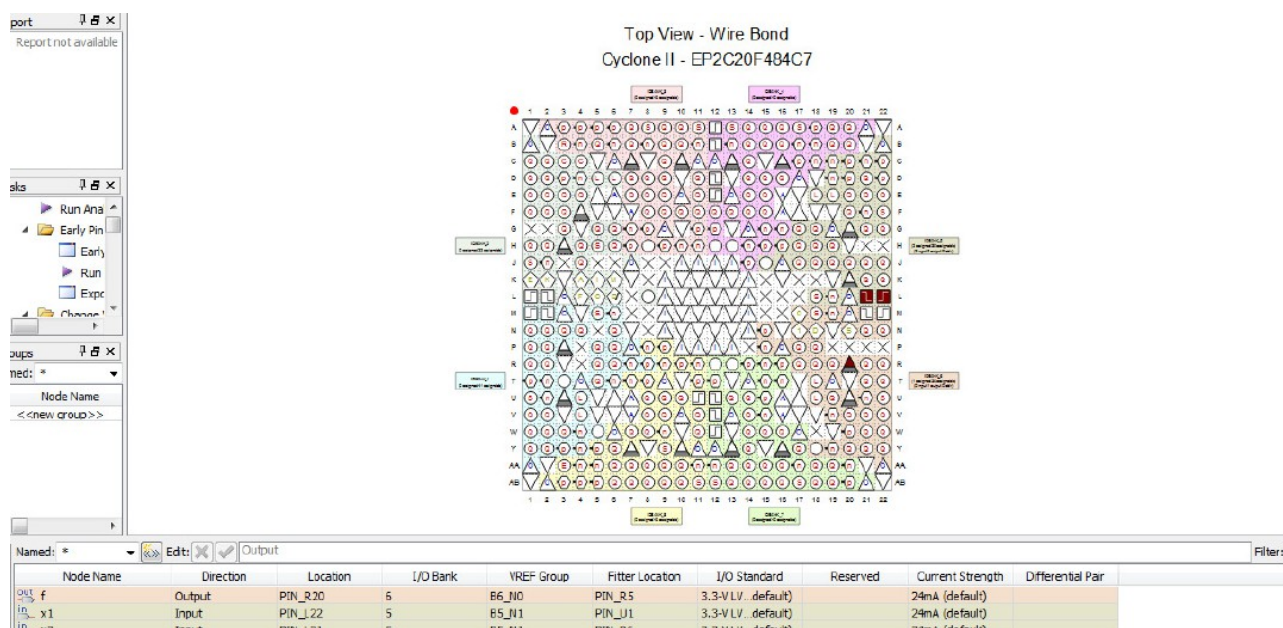
Приликом описа архитектуре кориштен је тзв. *Dataflow* опис, тј. кориштене су Булове једначине. Резултати за све могуће бриједности улазних промјенљивих дате су на следећој слици:



	Msgs														
lab1/x1	0														
lab1/x2	0														
lab1/f	0														

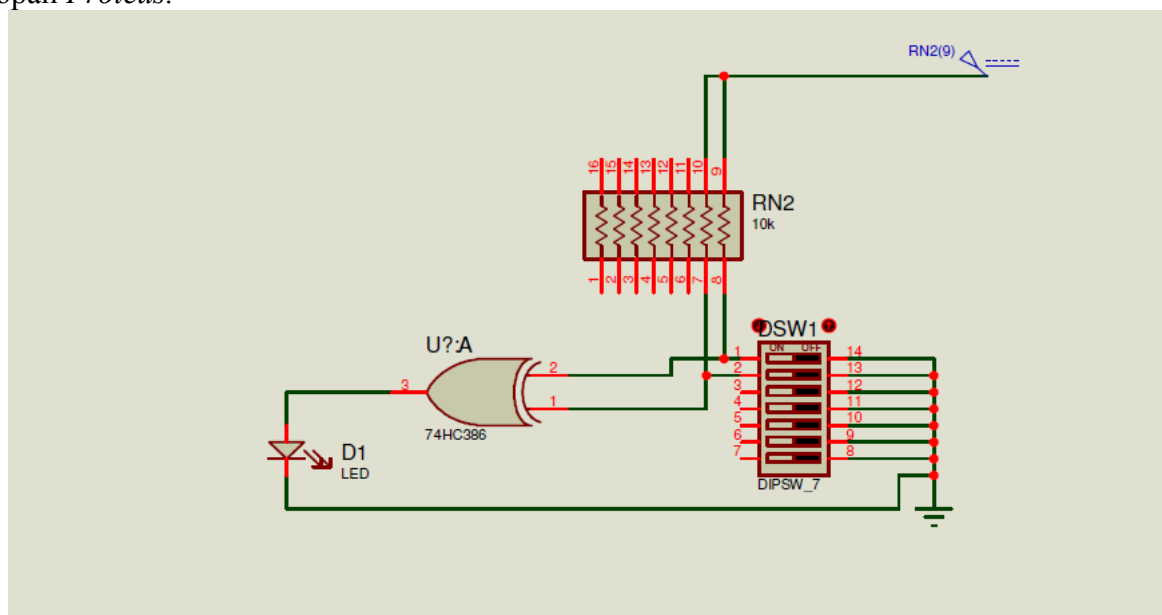
Сл. 1.2

Са претходне слике јасно је да је дата комбинациона мрежа двоулазно *xor* коло. На слици 1.3 приказано је повезивање модула са одговарајућим пиновима FPGA компоненте:



Сл. 1.3 Повезивање са одговарајућим пиновима FPGA компоненте

На слици 1.4 дата је електрична шема за тестирање дизајна. Умјесто RTL шеме одабран *Proteus*.



Сл. 1.4 Електрична шема за тестирање пројекта

Вјежба бр. 2

VHDL – Секвенцијална мрежа

1. Кратак опис вјежбе

Циљ ове вјежбе је реализовање система који се састоји из 3 модула:

1. Дјелитељ фреквенције
2. Машина стања којом је имплементиран BCD бројач
3. BCD->7 сегментни конвертор

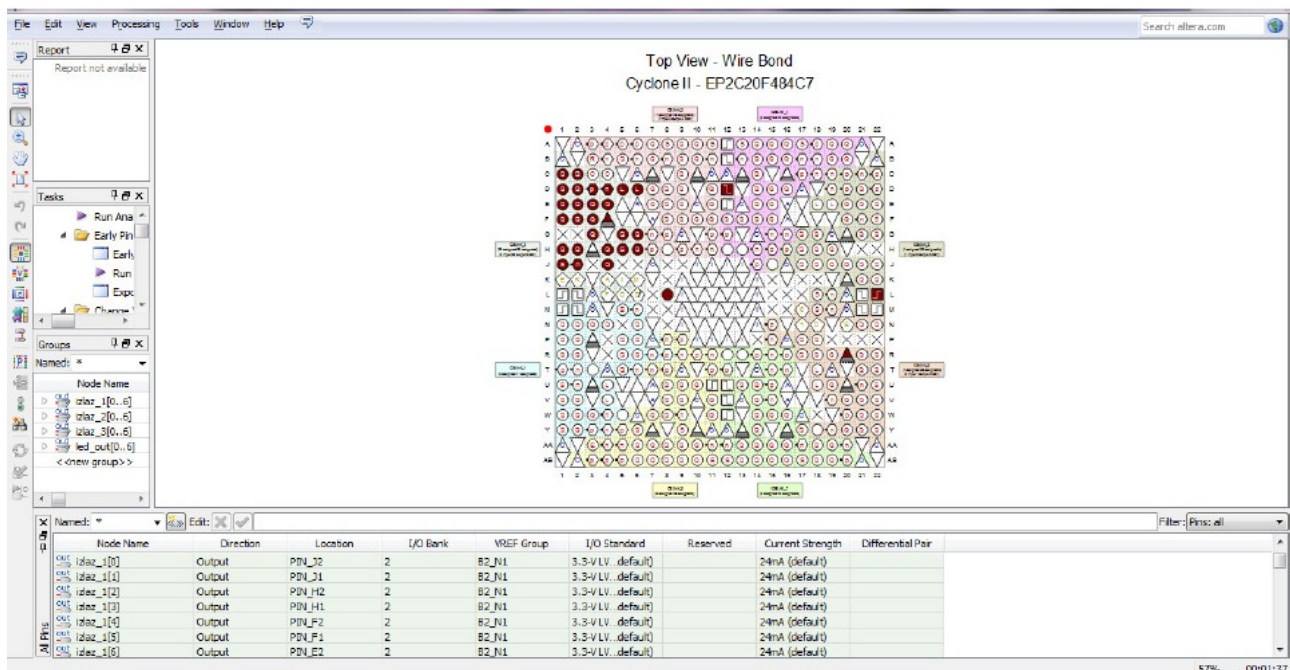
Повезивање модула је извршено у оквиру VHDL фајла *sistem*.

2. Кориштени софтвер и опрема

За реализацију је кориштен програмски пакет **Quartus II** произвођача Altera као и развојно окружење DE1 од истог произвођача.

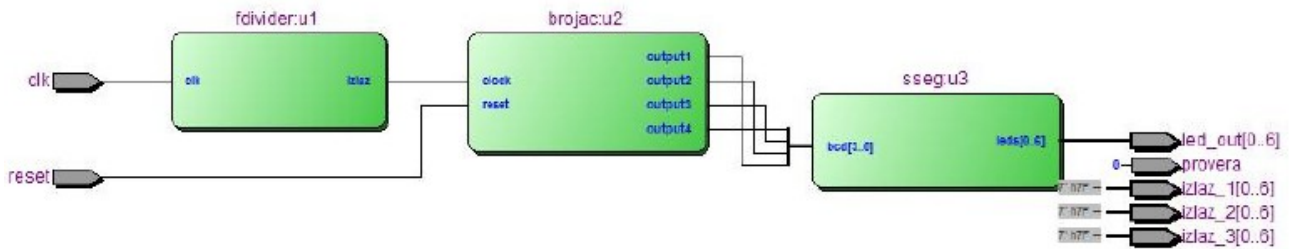
3. Резултати и закључак

Приликом описа архитектуре кориштен је концепт машине стања и то Мурове машине стања, јер се излази мијењају синхронно такту и независно од других улазних сигнала. На слици 2.1 приказано је повезивање са одговарајућим пиновима FPGA компоненте:



Сл. 2.1 Повезивање са одговарајућим пиновима FPGA компоненте

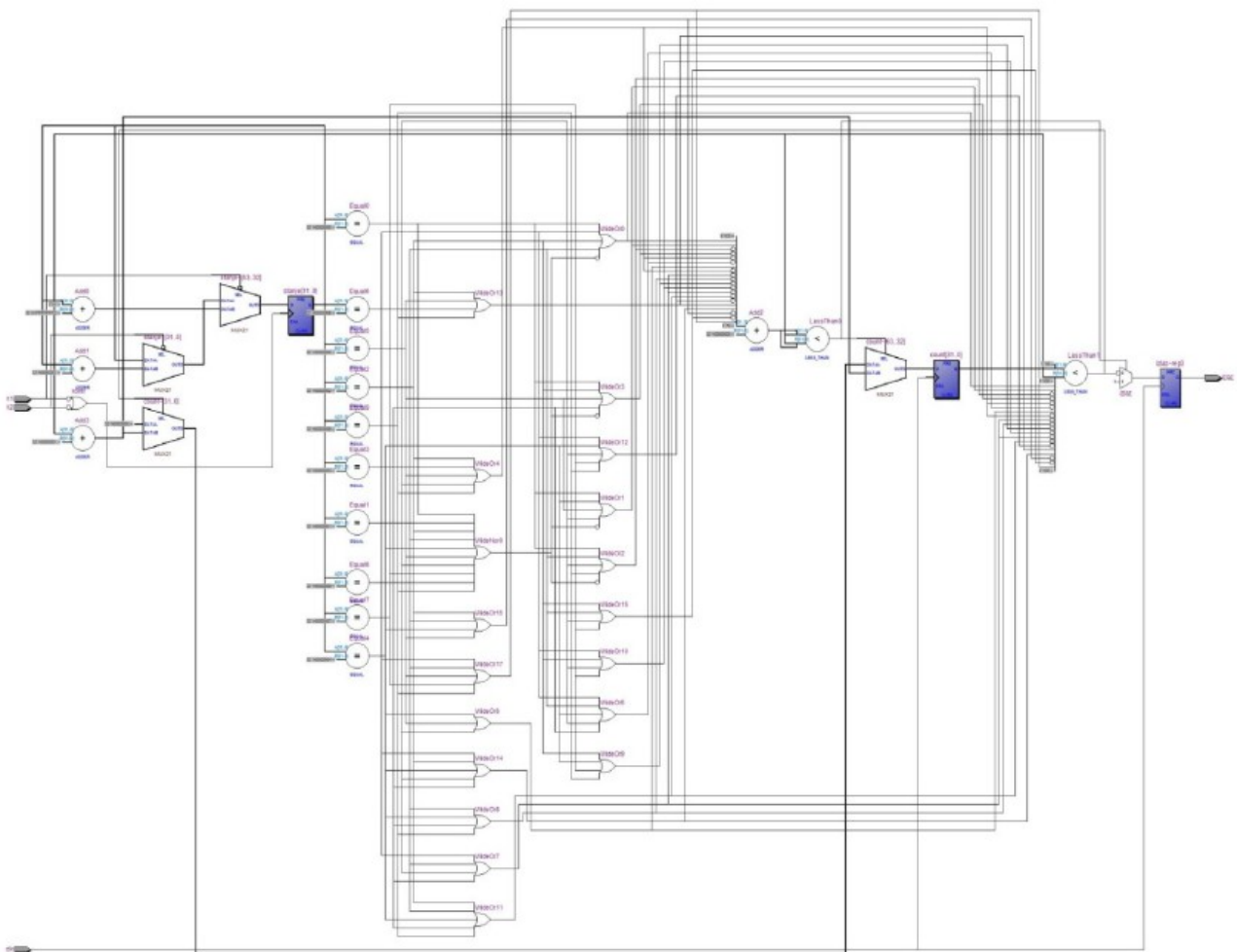
На слици 2.2 дата је логичка шема за тестирање дизајна.



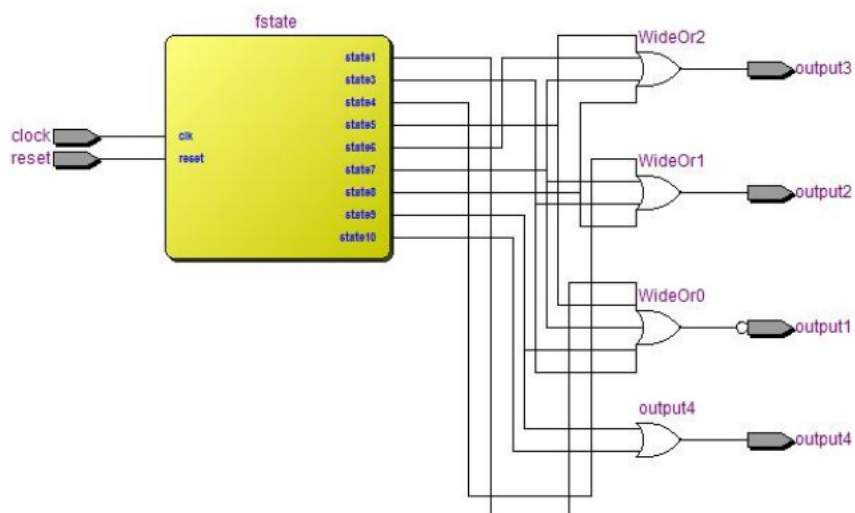
Сл. 2.2 Логичка шема комплетног система

Са претходне слике уочљива је модуларна структура система, као начин повезивања преко интерних сигнала. Излази система погоне четири седмосегментна дисплеја у формацији заједничка анода (стога су ЛЕД диоде активне на низак логички ниво).

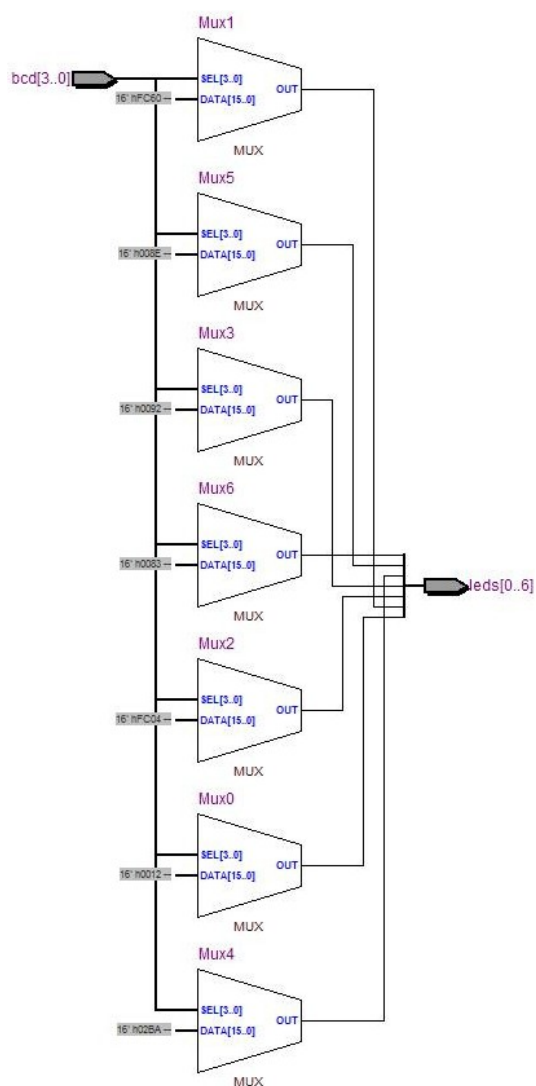
На наредне 3 слике дате су електричне шеме појединих модула.



Сл. 2.3 Електрична шема дјелитеља фреквенције

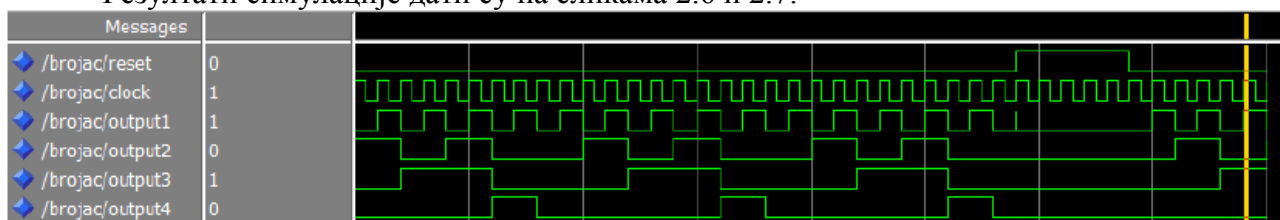


Сл. 2.4 Електрична шема бројача

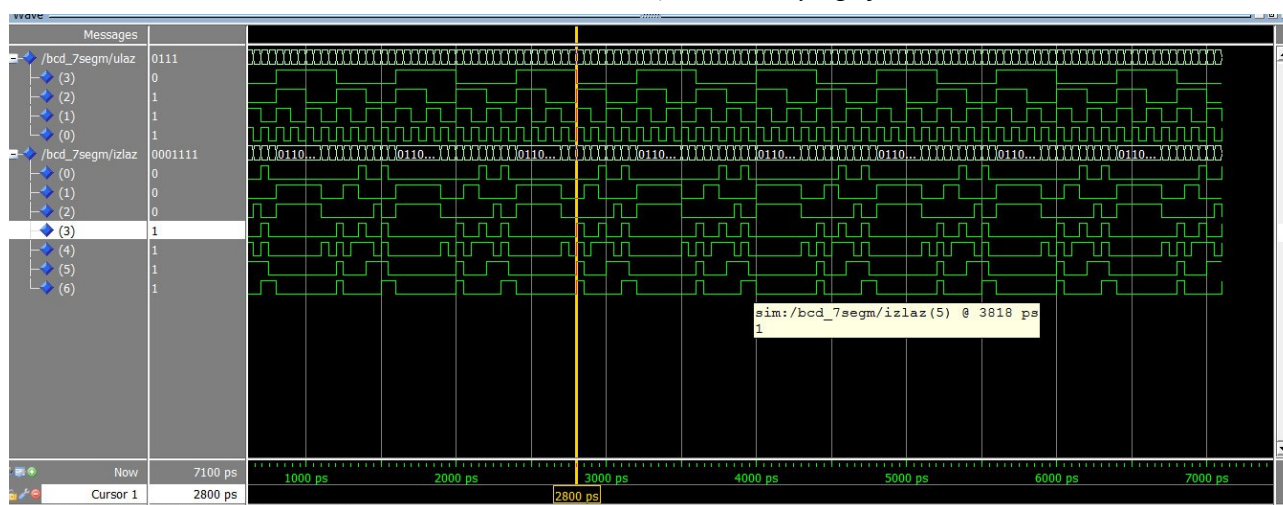


Сл. 2.5 Електрична шема *bcd_7seg* декодера

Резултати симулације дати су на сликама 2.6 и 2.7:



Сл. 2.6 Таласни облици на излазу бројача



Сл. 2.7 Таласни облици на излазу конвертора

Вјежба бр. 3

VHDL – Структурно пројектовање коришћењем шематског едитора

1. Кратак опис вјежбе

У овој вјежби упознаћемо се са шематским едитором који се налази у оквиру *Quartus II* софтвера и са основним елементима који се у оквиру овог софтверског алата налазе. Најприје ће бити описана методологија пројектовања шримјеном библиотеке примитивних логичких функција.

2. Кориштени софтвер и опрема

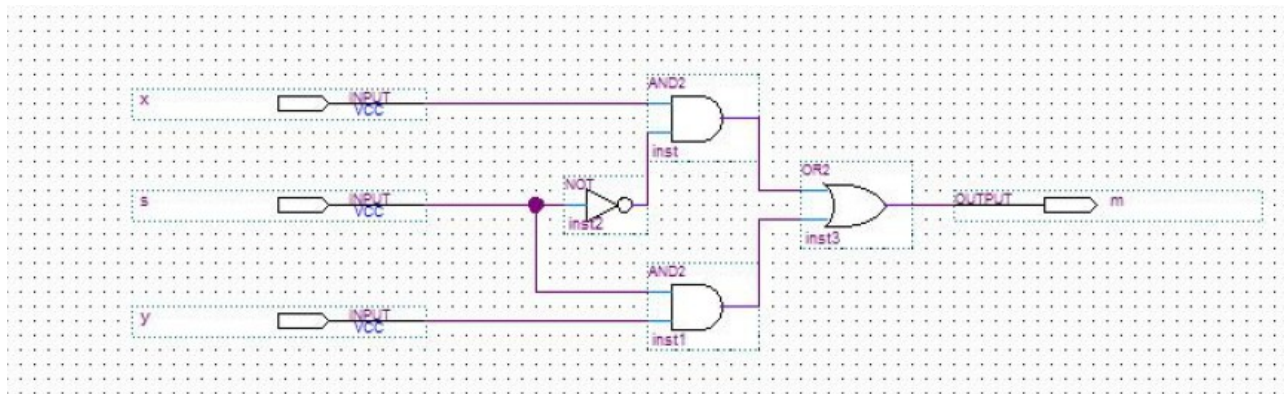
За реализацију је кориштен софтверски пакет *Quartus II* произвођача Altera као и развојно окружење DE1 од истог произвођача.

Табела 1. Кориштене компоненте

Компонента	Вриједност	Библиотека
Inst	///	Primitives/logic
Inst1	///	Primitives/logic
Inst2	///	Primitives/logic
Inst3	///	Primitives/logic
Input	///	Primitives/pin
Output	///	Primitives/logic

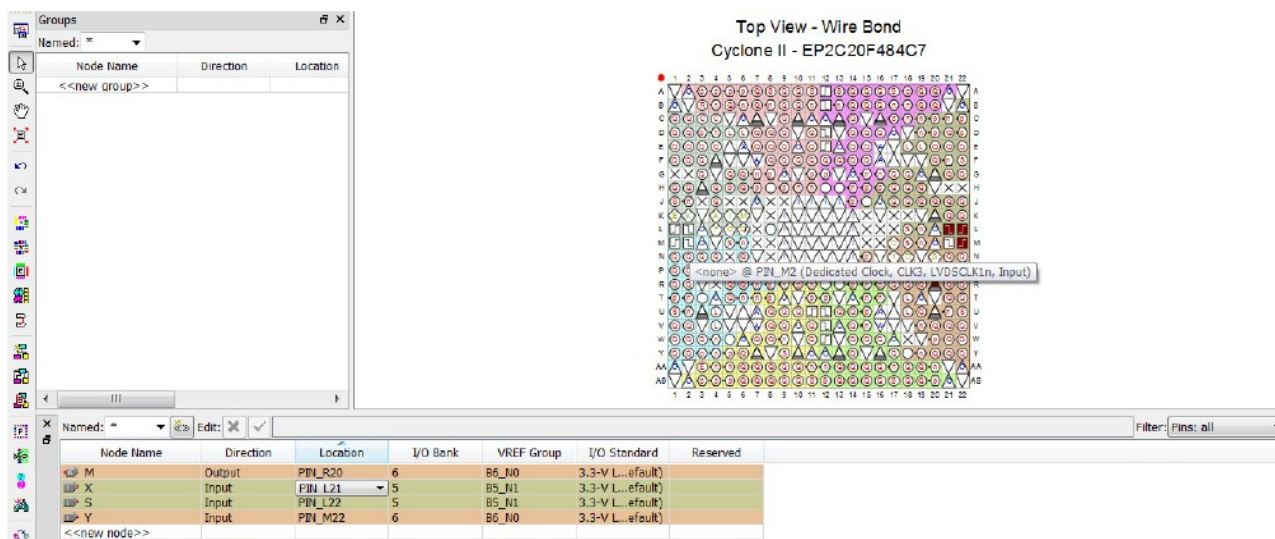
3. Резултати и закључак

У овом дијелу су израђени задаци за самостални рад. Прво је реализован мултиплексор 2/1 структурним стилем описа дизајна. Шематски дијаграм је приказан на сљедећој слици:

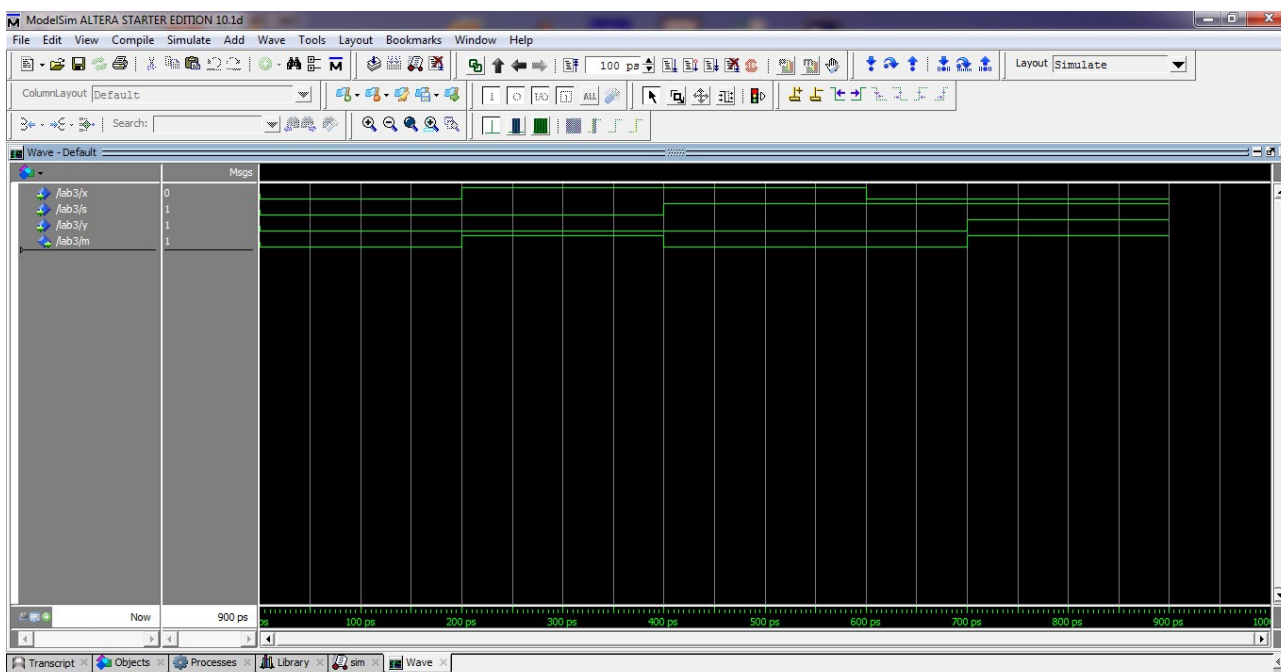


Сл. 3.1 Schematic diagram

На слици 3.2 дат је распоред додијелених пинова за претходни дизајн:

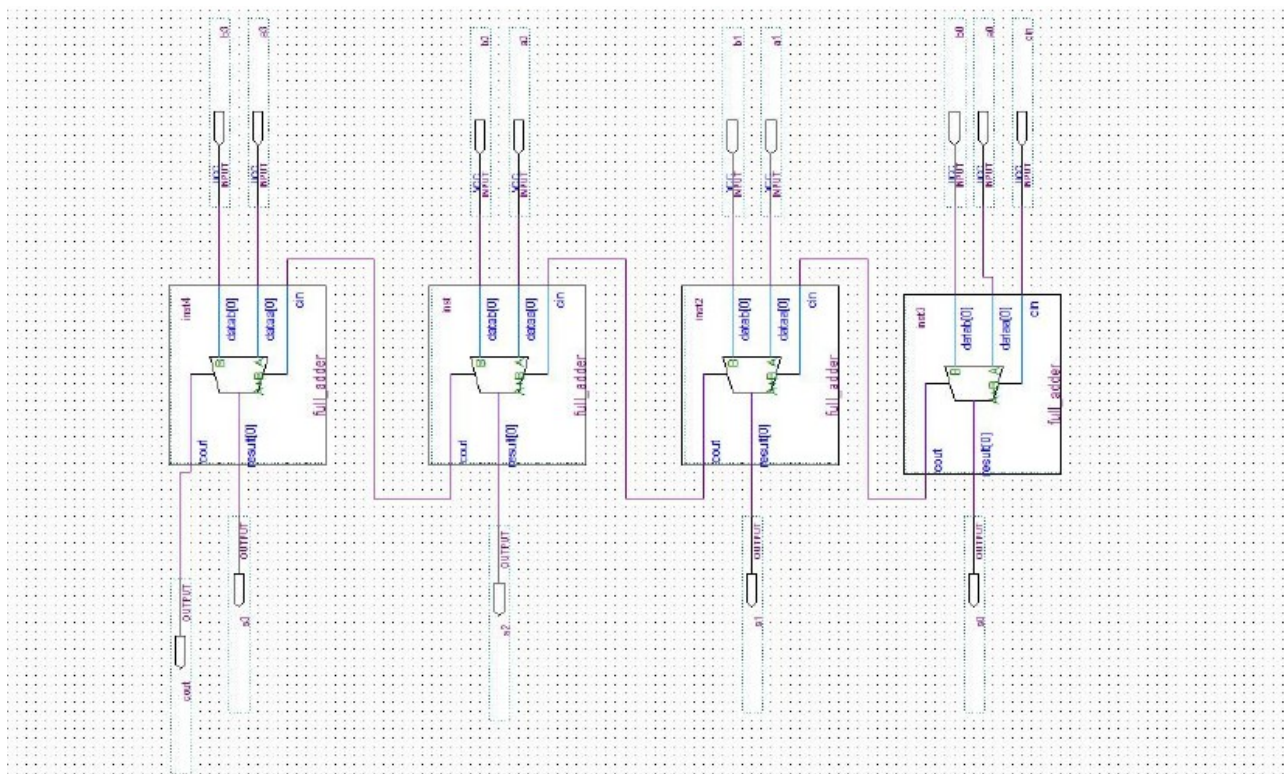


Сл. 3.2 Повезивање са одговарајућим пиновима *FPGA* компоненте



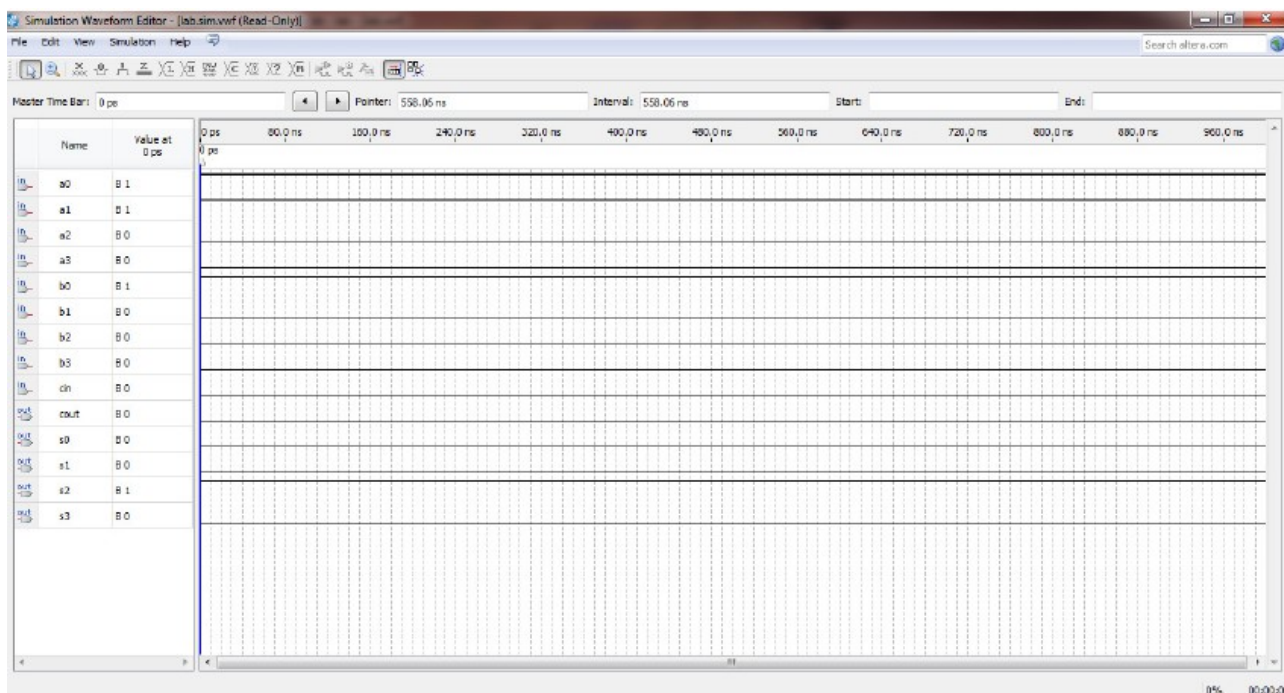
Сл. 3.3 Таласни облици улаза и излата мултиплексора 2/1

Затим је требало реализовати 4-битни сабирач кориштењем библиотеке параметризованих модула:

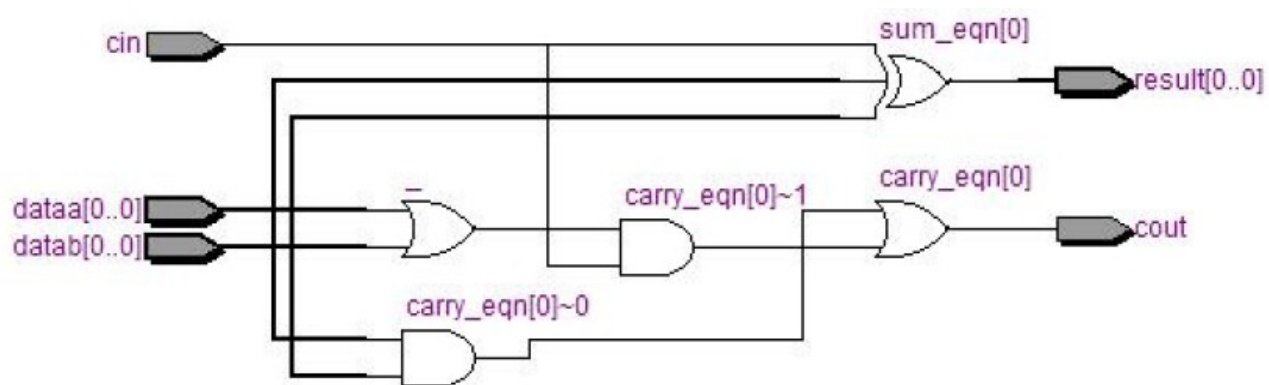


Сл. 3.4 Schematic diagram 4-битног сабирача

Резултати симулације дати су на слидећој слици (као примјер узето је $A=3$ $B=1$ $S=4$ без преноса):



Сл. 3.5 Таласни облици улаза и излаза 4-битног сабирача



Сл. 3.6 Електрична шема једнобитног потпуног сабирача

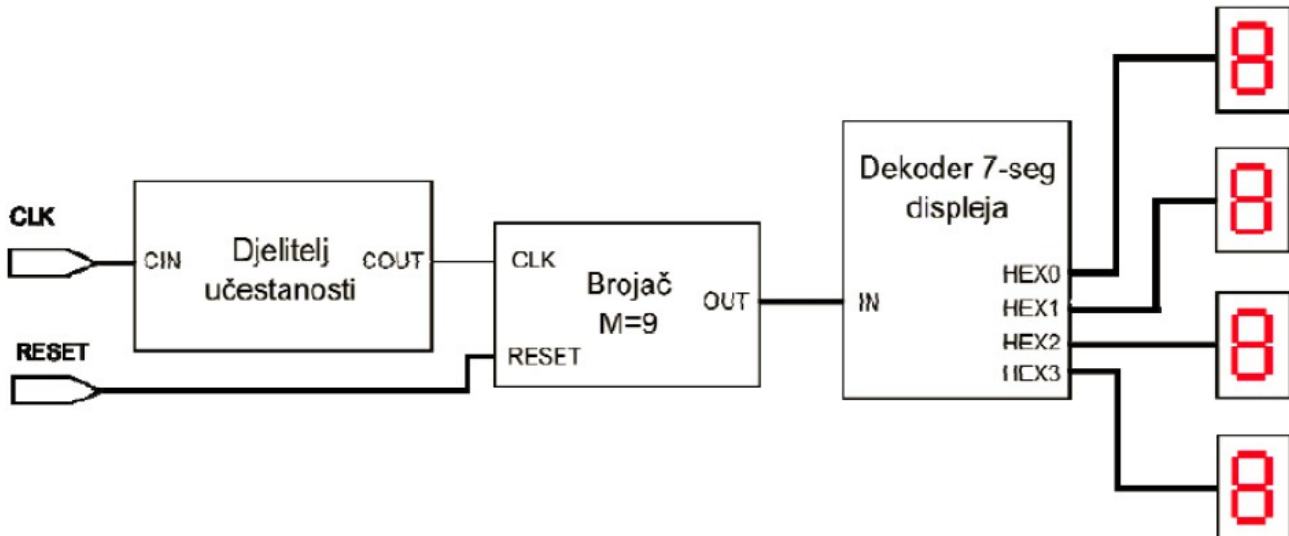
Вјежба бр. 4

VHDL – Помјерајући текст

1. Кратак опис вјежбе

Циљ ове лабораторијске вјежбе је пројектовање система за приказ поруке “HELLO” на 7-сегментним дисплејима у тзв. *Ticker Tape* начину рада (порука се приказује слово по слово с десна улијево) са једном секундом закашњења.

На сл. 1 дата је блок шема система:



Сл. 4.1 Блок-шема дигиталног система који је потребно реализовати

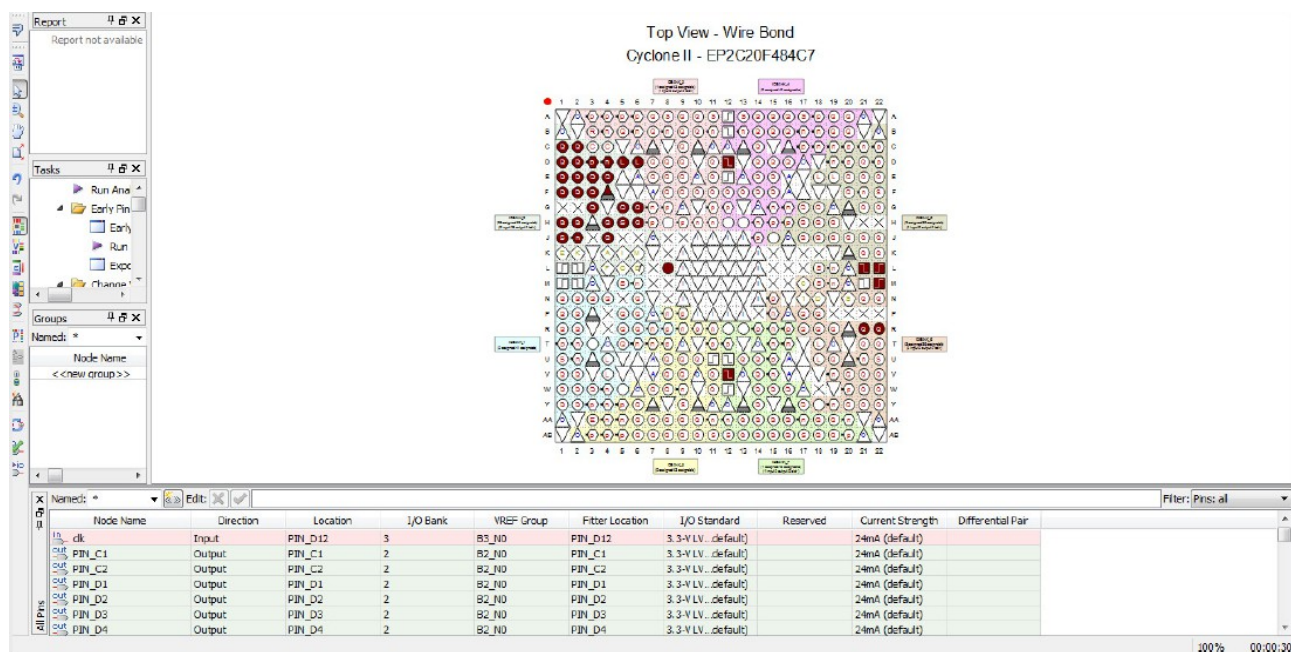
Потребно је реализовати три горе наведена модула и повезати их на структурном нивоу. VHDL фајлови који описују појединачне модуле дати су у склопу припреме за ову вјежбу. На основу VHDL фајлова генеришу се одговарајући блокови.

2. Кориштени софтвер и опрема

За реализацију је кориштен софтверски пакет *Quartus II* произвођача Altera као и развојно окружење DE1 од истог произвођача.

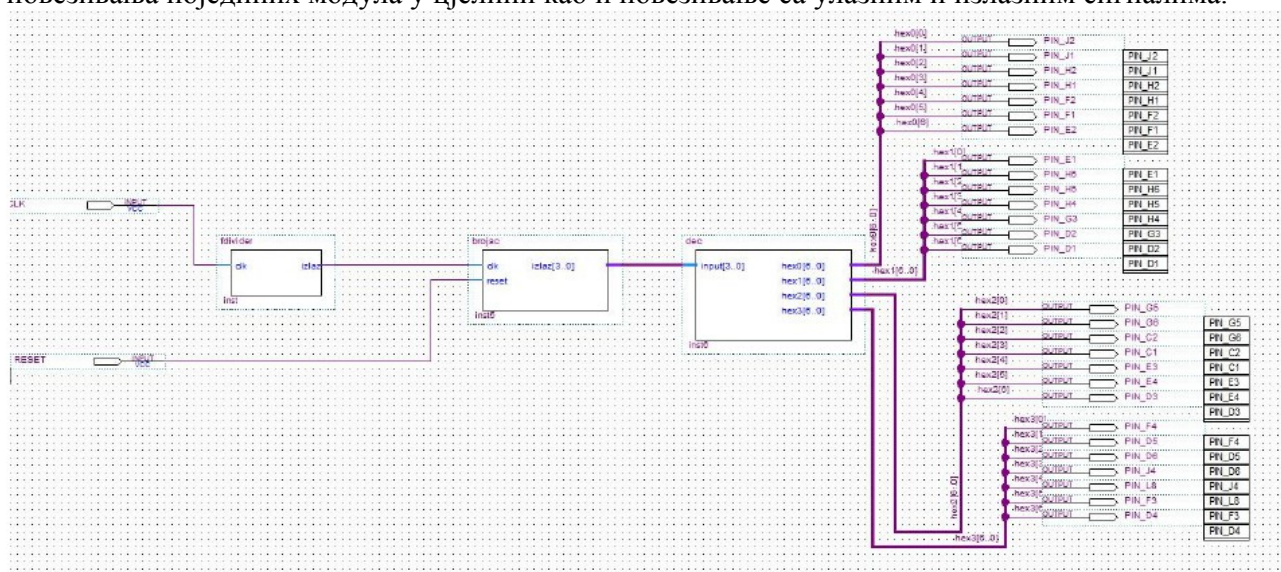
3. Резултати и закључак

На слици 4.2 дато је повезивање сигнала ентитета са пиновима FPGA компоненте са развојног окружења.



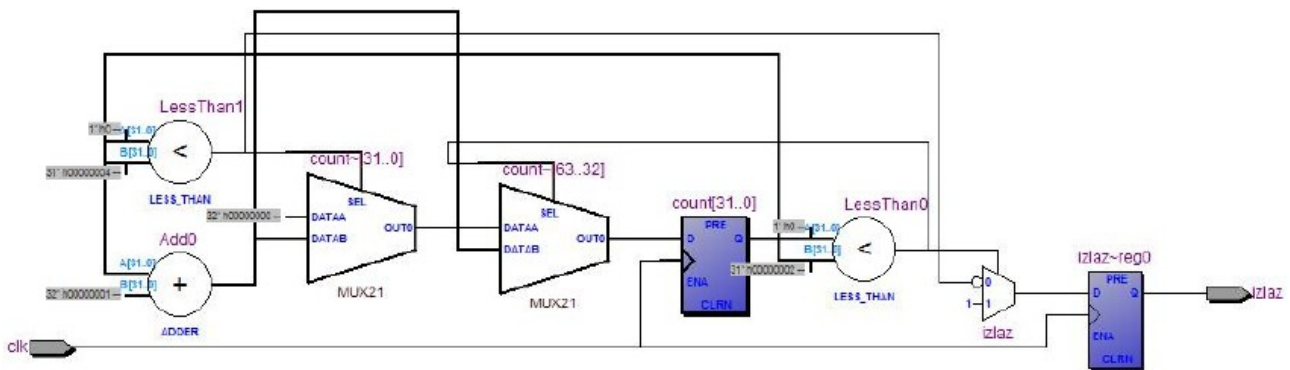
Сл. 4.2 Повезивање са одговарајућим пиновима FPGA компоненте

На наредној слици дата је блок шема система на структурном нивоу, гдје је видљив начин повезивања појединих модула у цјелини као и повезивање са улазним и излазним сигнаима.

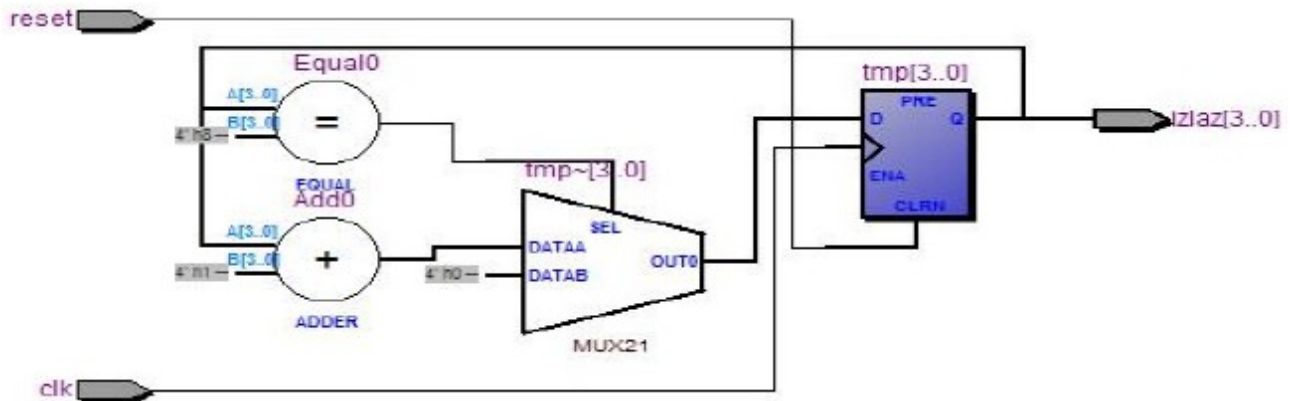


Сл. 4.3 Блок шема система

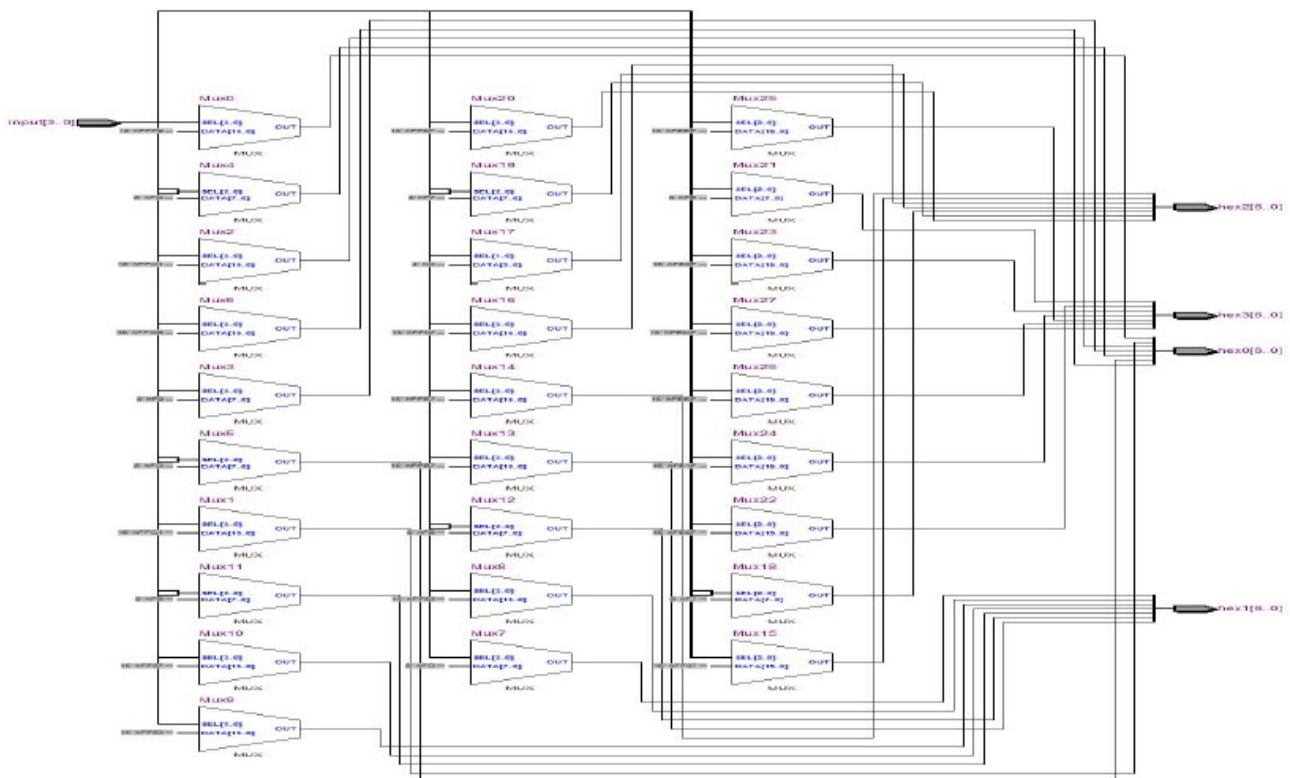
На сликама 4.4, 4.5 и 4.6 дата је RTL имплементација дизајна:



Сл. 4.4 Дјелитељ учестаности



Сл. 4.5 Бројач модула 5



Сл. 4.6 Декодер

Прва модификација система састоји се у томе да се брзина поруке мијења у опсегу од 0.1 де 1 секунде са корацима од 0.1 секунд. Брзина поруке одређена је вриједношћу прекидача SW3-0 (већој вриједности одговара већа брзина).

VHDL фајл је дат у листингу 4.1.

Листинг 1.1

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

entity brojac is
port(
    clk,reset:in std_logic;
    izlaz:out std_logic_vector(3 downto 0)
);
end brojac;

architecture ponasanje of brojac is
signal tmp:unsigned(3 downto 0):="0000";
begin
process(clk,reset)
begin
    if reset='1' then
        tmp<="0000";
    elsif rising_edge(clk) then
        if(tmp="1000") then
            tmp<="0000";
        else
            tmp<=tmp+1;
        end if;
    end if;
end process;

izlaz<=std_logic_vector(tmp);
end ponasanje;
```

```
library ieee;
use ieee.std_logic_1164.all;
entity fdivider is
port(
    clk,reset:in std_logic;
    izlaz:out std_logic;
    stanje:std_logic_vector(3 downto 0)
);
end fdivider;

architecture ponasanje of fdivider is
signal count:integer:=0;
signal var:integer:=27000000;
```



```
begin
process (stanje)
begin
    case stanje is
        when "0000"=> var<=27000000;
        when "0001"=> var<=24300000;
        when "0010"=> var<=21600000;
        when "0011"=> var<=18900000;
        when "0100"=> var<=16200000;
        when "0101"=> var<=13500000;
        when "0110"=> var<=10800000;
        when "0111"=> var<=8100000;
        when "1000"=> var<=5400000;
        when "1001"=> var<=2700000;
        when others=> var<=27000000;
    end case;
end process;
```

```
process(clk,reset)
begin
    if reset='1' then
        izlaz<='0';
        count<=0;
    elsif rising_edge(clk)then
        if count<var/2 then
            izlaz<='1';
            count<=count+1;
        elsif count<var then
            izlaz<='0';
            count<=count+1;
        else
            count<=0;
            count<=0;
            izlaz<='1';
        end if;
    end if;
end process;
end ponasanje;
```

```
library ieee;
use ieee.std_logic_1164.all;
entity dec is
port(
    input : in std_logic_vector (3 downto 0);
    hex0, hex1, hex2, hex3 : out std_logic_vector(6 downto 0)
);
end dec;

architecture behav_dec of dec is
begin
```

```
with input select
    hex0 <= "1111111" when "0000",
    "0001001" when "0001",
    "0000110" when "0010",
    "1000111" when "0011",
    "1000111" when "0100",
    "1000000" when "0101",
    "1111111" when "0110",
    "1111111" when "0111",
    "1111111" when "1000",
    "1111111" when others;

with input select
    hex1 <= "1111111" when "0000",
    "1111111" when "0001",
    "0001001" when "0010",
    "0000110" when "0011",
    "1000111" when "0100",
    "1000111" when "0101",
    "1000000" when "0110",
    "1111111" when "0111",
    "1111111" when "1000",
    "1111111" when others;

with input select
    hex1 <= "1111111" when "0000",
    "1111111" when "0001",
    "0001001" when "0010",
    "0000110" when "0011",
    "1000111" when "0100",
    "1000111" when "0101",
    "1000000" when "0110",
    "1111111" when "0111",
    "1111111" when "1000",
    "1111111" when others;

with input select
    hex2 <= "1111111" when "0000",
    "1111111" when "0001",
    "1111111" when "0010",
    "0001001" when "0011",
    "0000110" when "0100",
    "1000111" when "0101",
    "1000111" when "0110",
    "1000000" when "0111",
    "1111111" when "1000",
    "1111111" when others;

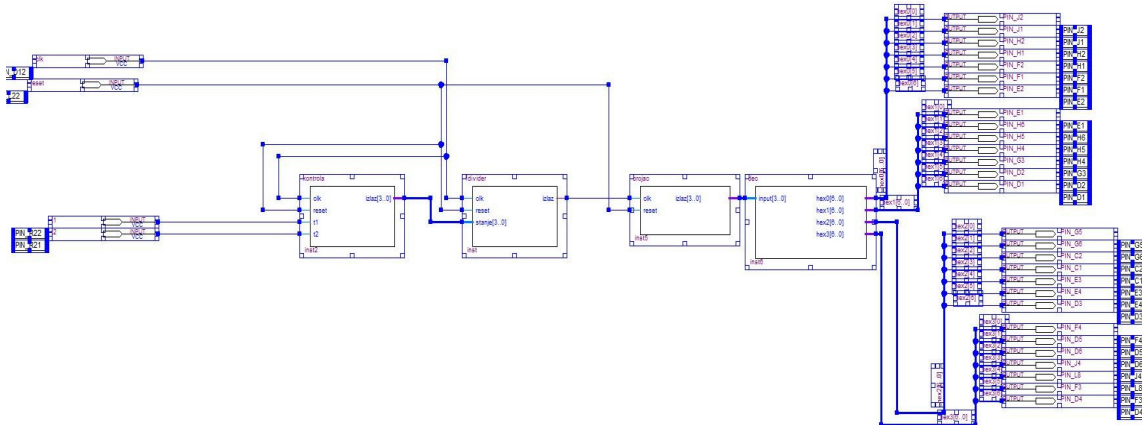
with input select
    hex3 <= "1111111" when "0000",
    "1111111" when "0001",
    "1111111" when "0010",
    "1111111" when "0011",
    "0001001" when "0100",
```

```

"0000110" when "0101",
"1000111" when "0110",
"1000111" when "0111",
"1000000" when "1000",
"1111111" when others;
end behav_dec;

```

Друга модификација је у случају да се брзина порука мијења тастерима за убрзавање и успоравање:



Сл. 4.7 Блок шема модификованог система

VHDL код је идентичан горе наведеном коду осим додатног система *kontrola* у којем је мијењање брзине притиском тастера ријешено употребом машине стања.

```

library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
entity kontrola is
port(
    clk,reset,t1,t2:in std_logic;
    izlaz:out std_logic_vector(3 downto 0)
);
end kontrola;

architecture ponasanje of kontrola is
type state is(s0,s1,s2,s3);
signal cstate,nstate:state;
signal temp:unsigned(3 downto 0):="0000";
begin
process(clk,reset)
begin
    if reset='1' then
        cstate<=s0;
        izlaz<=(others=>'0');
    elsif rising_edge(clk) then
        cstate<=nstate;
        izlaz<=std_logic_vector(temp);
    end if;
end process;

```

```
        end if;
end process;

process(cstate,t1,t2)
begin
    case cstate is
        when s0=>
            if t1='0' then
                nstate<=s1;
            elsif t2='0' then
                nstate<=s2;
            else
                nstate<=s0;
            end if;
        when s1=>
            temp<=temp+1;
            if temp=11 then
                temp<=10;
            end if;
            nstate<=s3;
        when s2=>
            if temp>0 then
                temp<=temp-1;
            end if;
            nstate<=s3;
        when s3=>
            if t1='1' and t2='1' then
                nstate<=s0;
            else
                nstate<=s3;
            end if;
        end case;
end process;
end behaviour;
```

Вјежба бр. 5

VHDL – Машина стања

1. Кратак опис вјежбе

Опис поступка пројектовања дигиталног система заснованог на стандардним секвенцијалним мрежама илустроваће се примјером система за приказ поруке “HELLO” на 7-сегментним дисплејима који је идентичан систему који је описан у лабораторијској вјежби 4. Једина разлика је у томе што ће се опис система извршити одговарајућим дијаграмом машине са коначним бројем стања у *State Machine Editor* алату који је доступан у оквиру *Quartus II* софтвера.

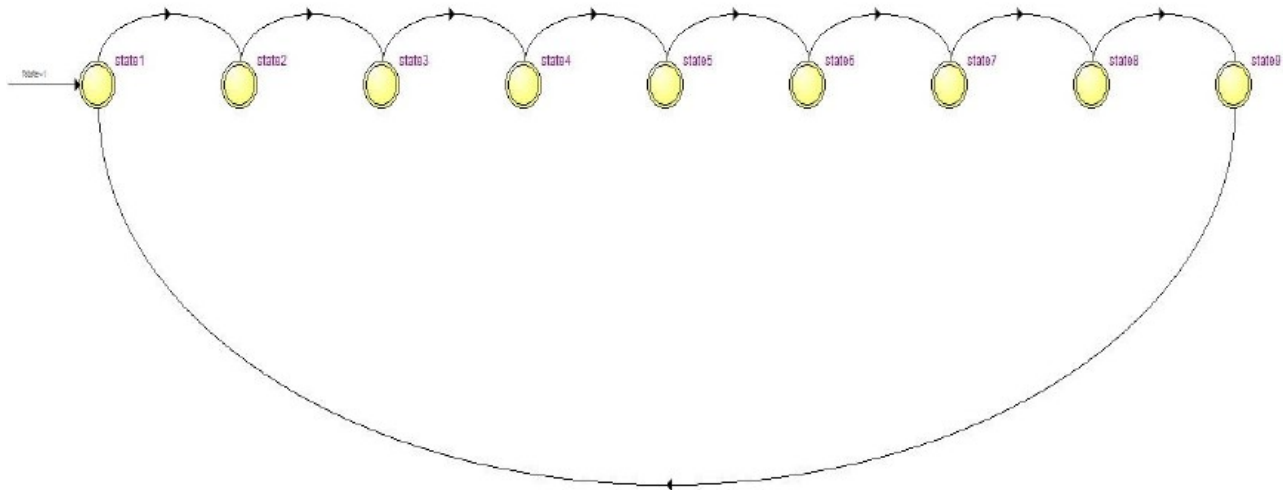
Дигитални систем има исте карактеристике као и систем описан у лабораторијској вјежби 4. Једина разлика је у томе што се за реализацију не користе бројач модула 9 (за 9 различитих стања мреже) и декодер за 7-сегментне дисплеје, већ се реализација система своди на пројектовање јединственог контролера система, заснованог на машини стања, у који су укључене све функционалности система.

2. Кориштени софтвер и опрема

За реализацију је кориштен софтверски пакет *Quartus II* произвођача Altera као и развојно окружење DE1 од истог произвођача.

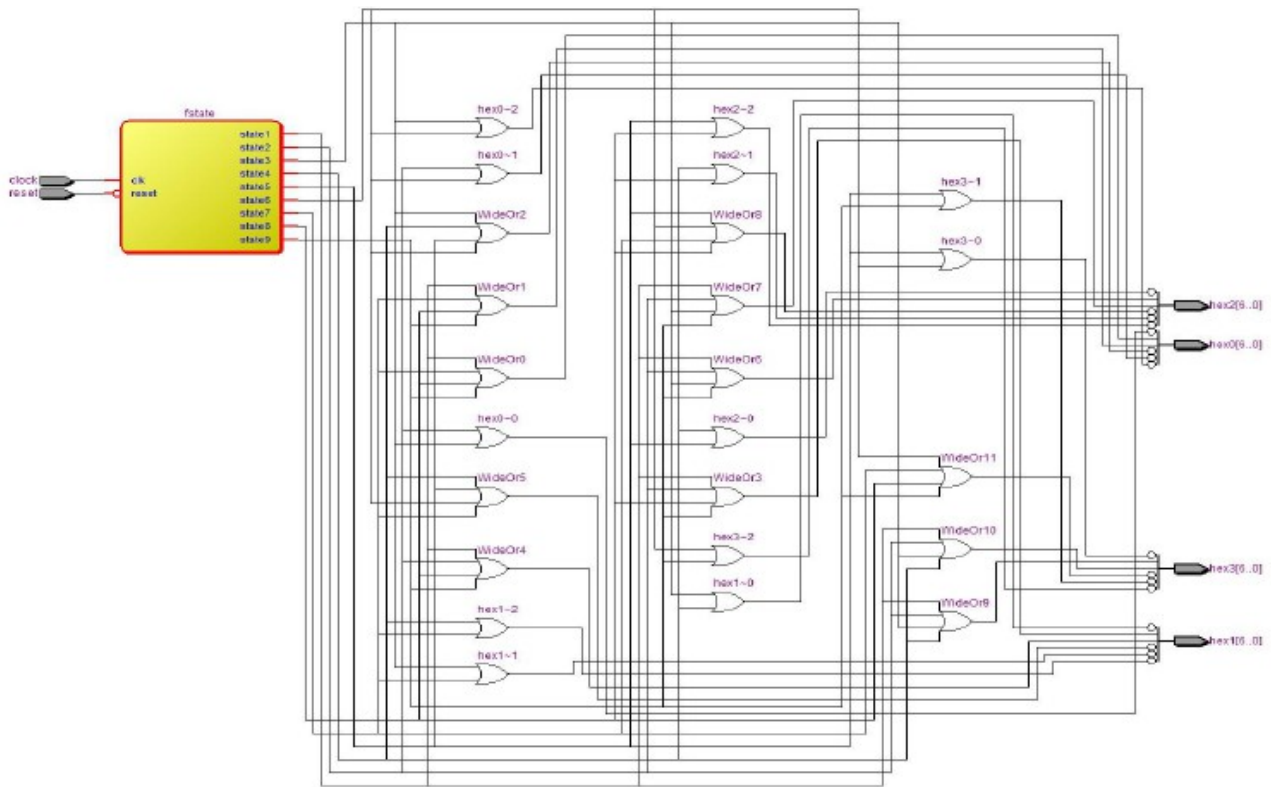
3. Резултати и закључак

На слици 5.1 дат је изглед добијене машине стања имплементираног дизајна:



Сл. 5.1 Машина стања (FSM Chart)

Јасна је структура машине стања која имплементира бројач модула 9. На наредној слици дата је електрична шема (RTL имплементација) дизајна:



Сл. 5.2 RTL имплементација дизајна

У оквиру наредног листинга дат је генерисани VHDL код машине стања.

Листинг 5.1

```

LIBRARY ieee;
USE ieee.std_logic_1164.all;
ENTITY controller IS
PORT (
    reset : IN STD_LOGIC := '0';
    clock : IN STD_LOGIC;
    hex0:out std_logic_vector(6 downto 0);
    hex1:out std_logic_vector(6 downto 0);
    hex2:out std_logic_vector(6 downto 0);
    hex3:out std_logic_vector(6 downto 0)
);
END controller;

ARCHITECTURE BEHAVIOR OF controller IS
TYPE type_fstate IS (state1,state2,state3,state4,state5,state6,state7,state8,state9);
SIGNAL fstate : type_fstate;
SIGNAL reg_fstate : type_fstate;
BEGIN
PROCESS (clock,reset,reg_fstate)
BEGIN
    IF (reset='0') THEN
        fstate <= state1;
    ELSIF (clock='1' AND clock'event) THEN
        fstate <= reg_fstate;
    END IF;

```

```

END PROCESS;

PROCESS (fstate)
BEGIN
    CASE fstate IS
        WHEN state1 =>
            hex0<="1111111";
            hex1<="1111111";
            hex2<="1111111";
            hex3<="1111111";
            reg_fstate <= state2;
        WHEN state2 =>
            hex0<="0001001";
            hex1<="1111111";
            hex2<="1111111";
            hex3<="1111111";
            reg_fstate <= state3;
        WHEN state3 =>
            hex0<="0000110";
            hex1<="0001001";
            hex2<="1111111";
            hex3<="1111111";
            reg_fstate <= state4;
        WHEN state4 =>
            hex0<="1000111";
            hex1<="0000110";
            hex2<="0001001";
            hex3<="1111111";
            reg_fstate <= state5;
        WHEN state5 =>
            hex0<="1000111";
            hex1<="1000111";
            hex2<="0000110";
            hex3<="0001001";
            reg_fstate <= state6;
        WHEN state6 =>
            hex0<="1000000";
            hex1<="1000111";
            hex2<="1000111";
            hex3<="0000110";
            reg_fstate <= state7;
        WHEN state7 =>
            hex0<="1111111";
            hex1<="1000000";
            hex2<="1000111";
            hex3<="1000111";
            reg_fstate <= state8;
        WHEN state8 =>
            hex0<="1111111";
            hex1<="1111111";
    
```

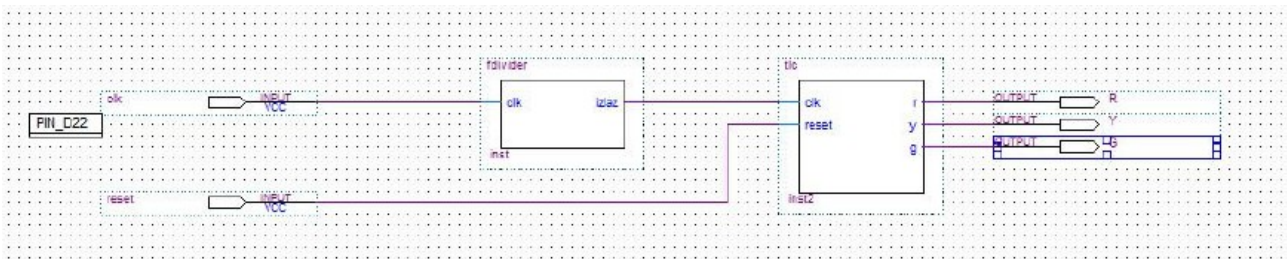


```

        hex2<="1000000";
        hex3<="1000111";
        reg_fstate <= state9;
    WHEN state9 =>
        hex0<="1111111";
        hex1<="1111111";
        hex2<="1111111";
        hex3<="1000000";
        reg_fstate <= state1;
    WHEN OTHERS =>
        hex0<="ZZZZZZZ";
        hex1<="ZZZZZZZ";
        hex2<="ZZZZZZZ";
        hex3<="ZZZZZZZ";
        report "Reach undefined state";
    END CASE;
END PROCESS;
END BEHAVIOR;
    
```

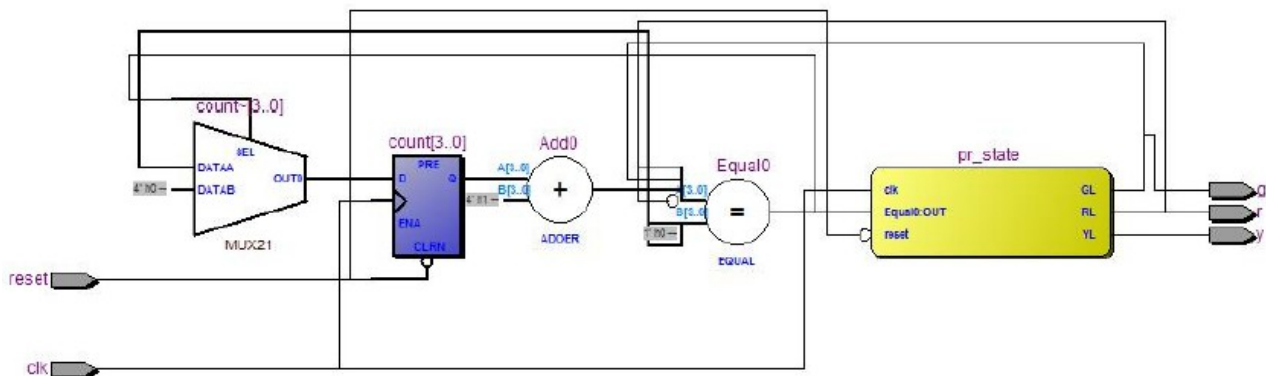
У оквиру задатка за самосталан рад требало је пројектовати контролер семафора коришћењем методологије машине стања. Као сијалице семафора треба да послуже диоде доступне на плочи DE1. Трајање зеленог свјетла треба да буде 5 секунди, жутог 1 секунда и црвеног 8 секунди. Обезбиједити одговарајућа времена коришћењем једног од извора такт-сигнала са плоче уз неопходне дјелитеље учестаности. Такође, треба обезбиједити асинхрони ресет система помоћу тастера KEY0 (претпоставити да је почетно стање система црвено свјетло).

Модул је назван TLC (*Traffic Light Controller*). Блок шема система приказана је на сл. 5.1:



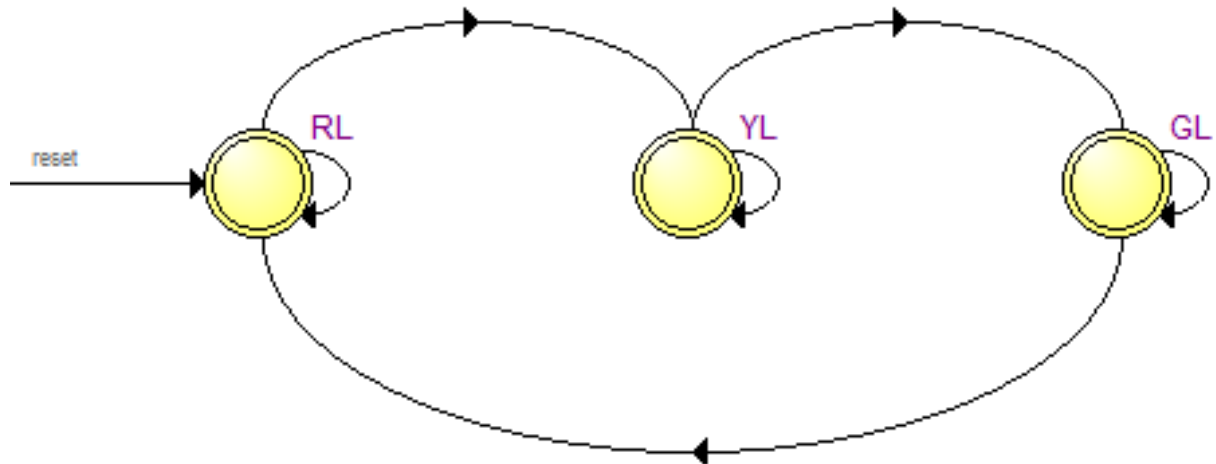
Сл. 5.3 Schematic Diagram траженог контролера (TLC-Traffic Light Controller)

На следећој слици дата је електрична шема система са машином стања:



Сл. 5.4 Електрична шема дизајна

На слици 5.5 дата је машина стања којом је реализован сам контролер:



Сл. 5.5 Машина стања

Реализација машине стања је крајње једноставна. Контролер остаје у одређеном стању док не протекне одређени временски интервал. Фреквенција такта који се доводи машини стања је претходно подијељена у дјелитељу фреквенције на вриједност 1 Hz.

VHDL код којим је имплементирана машина стања дат је у сљедећем листингу.

Листинг 5.2

```

LIBRARY ieee;
USE ieee.std_logic_1164.all;
ENTITY tlc IS
PORT (
    clk, reset: IN STD_LOGIC;
    r,y,g: OUT STD_LOGIC
); --izlazi: red,yellow,green
END tlc;

ARCHITECTURE behavior OF tlc IS
CONSTANT timeR : INTEGER := 8; --konstante koje predstavljaju trajanje pojedinih
--svjetala
CONSTANT timeY : INTEGER := 1;
CONSTANT timeG : INTEGER :=5;
TYPE state IS (RL,YL,GL);
SIGNAL pr_state, nx_state: state;
SIGNAL time : INTEGER RANGE 0 TO timeR; --lokalna varijabla

BEGIN
PROCESS (clk,reset)
VARIABLE count : INTEGER RANGE 0 TO timeR:=0; --brojac
BEGIN
    IF (reset='0') THEN
        pr_state <= RL;
        count := 0;
    ELSIF (clk'EVENT AND clk='1') THEN
        count := count + 1;
    
```

```
        IF (count = time) THEN
            pr_state <= nx_state;
            count := 0;
        END IF;
    END IF;
END PROCESS;

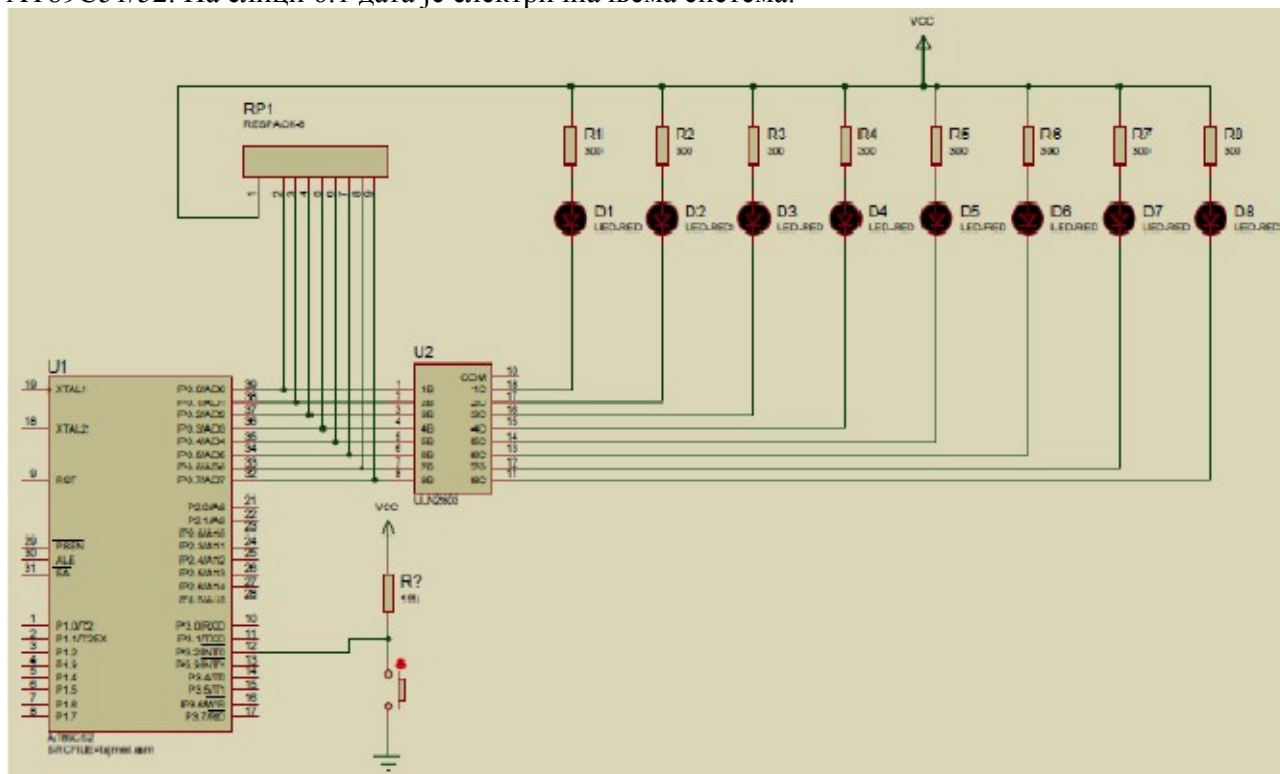
PROCESS (pr_state)
begin
    case pr_state is
        when RL=>
            r<='1';
            y<='0';
            g<='0';
            time<=timeR; --azuriranje vrijednosti lokalne varijable
            nx_state<=YL;
        when YL=>
            r<='0';
            y<='1';
            g<='0';
            time<=timeY;
            nx_state<=GL;
        when GL=>
            r<='0';
            y<='0';
            g<='1';
            time<=timeG;
            nx_state<=RL;
        end case;
    end process;
end behavior;
```

Вјежба бр. 6

МИКРОКОНТРОЛЕР AT89C51/52 – ТАЈМЕР

1. Кратак опис вјежбе

Циљ ове вјежбе је сигнализација преко 8 LED диода посредством микроконтролера AT89C51/52. На слици 6.1 дата је електрична шема система.



Сл. 6.1 Електрична шема система

Као што се може видјети са претходне слике, микроконтролер не може директно да се веже путем пинова за диоде због недовољног струјног капацитета (и то струјног капацитета по порту, а не пину). Кориштени су пинови порта 0 који нема интерне *pull-up* отпорнике (већ транзисторе са отвореним дрејном), па је неопходно користити екстерне *pull-up* отпорнике за дефинисање логичке 1. Са друге стране, за укључивање диода користи се драјверско коло ULN2803 које представља Дарлингтонову спрегу транзистора. Отпорници везани редно се прорачунавају на основу жељене вриједности струје диода ($\sim 10\text{mA}$).

2. Кориштени софтвер и опрема

Кориштен је софтвер *MIDE-51* (за асемблирање) и софтверски пакет *Proteus* (за симулацију). За практичну верификацију кориштено је развојно окружење *Easy8051*.

3. Резултати и симулација

Приликом модификације асемблерског кода из припреме додано је неколико инструкција које реалитују тзв. трчеће свјетло. Модификација је урађена у оквиру прекидне рутине тајмера 0, а састоји се у ротирању садржаја акумулатора (који је иницијално постављен на вриједност 01h). Асемблерски програм дат је у оквиру Листинга 6.1.

Листинг 6.1

```

DSEG ; segment podataka u internom RAM-u
    org 20h ; pocinje od adrese 20h

CSEG ; segment koda
org 0000h ; reset
    jmp POCETAK
org 0003h ; spoljasnji prekid 0
    reti
org 000bh ; prekid tajmera 0
    jmp PREKID_T0
org 0013h ; spoljasnji prekid 1
    reti
org 001bh ; prekid tajmera 1
    reti
org 0023h ; prekid serijskog porta
    reti
org 002bh ; prekid tajmera 2 kod 8052
    reti

org 0030h ; adresa pocetaka glavnog programa
POCETAK:
    mov p0 , #01h ; LED diode na portu 0 se gase
    mov p1 , #0FFh ; LED diode na portu 1 se gase
    mov p2 , #0FFh ; LED diode na portu 2 se gase
    mov p3 , #0FFh ; LED diode na portu 3 se gase
    mov r0 , #00h
    mov tmod , #01h ; tajmer T0 radi u modu 1
    mov ie , #82h ; omogucenje svih prekida
    mov th0 , #3Ch ; u tajmer 0 D8F0
    mov tl0 , #0B0h ; sto odgovara prekidu tajmera svakih 50ms na 12MHz
    setb tr0 ; startovanje tajmera 0

GLAVNI:
    jmp GLAVNI ; vracamo se na pocetak programa

PREKID_T0:
    clr tr0
    inc r0
    cjne r0 , #20, PREKID_T0_1
    mov r0 , #00h
    mov a , p0 ;modifikacija programa
    rl a ;"trcece" svjetlo
    mov p0 , a
PREKID_T0_1:
    mov th0 , #3Ch;
    mov tl0 , #0B0h
    setb tr0

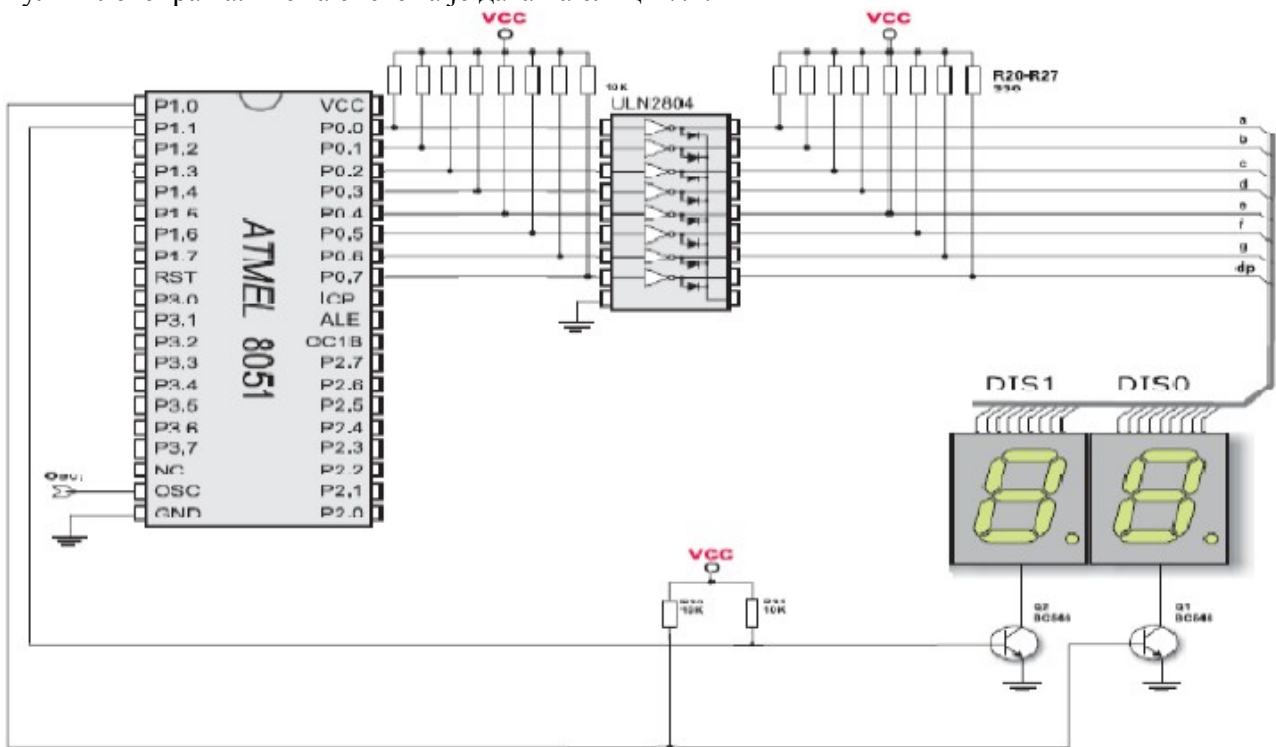
RETI
END
    
```

Вјежба бр. 7

УПРАВЉАЊЕ 7с ДИСПЛЕЈЕМ

1. Кратак опис вјежбе

Циљ ове вјежбе је управљање са два 7-сегментна дисплеја и то у режиму тзв. мултиплексирања. Шема система је дата на слици 7.1.



Сл. 7.1 Електрична шема система

На шеми је потребно уочити неколико ствари. Прво, употреједљене су три групе *pull-up* отпорника. Горња група од 10k дефинишу стање логичке 1 на пиновима на порту 0 због *open-drain* излаза.

Доња група од 10k служи да повећа базну струју селекционих транзистора Q1 и Q2 обезбјеђујући њихово засићење (порт 1 посједује интерне *pull-up* отпорнике).

Трећа група дефинише стање логичке 1 (излази струјног бафера су тада у стању високе импедансе).

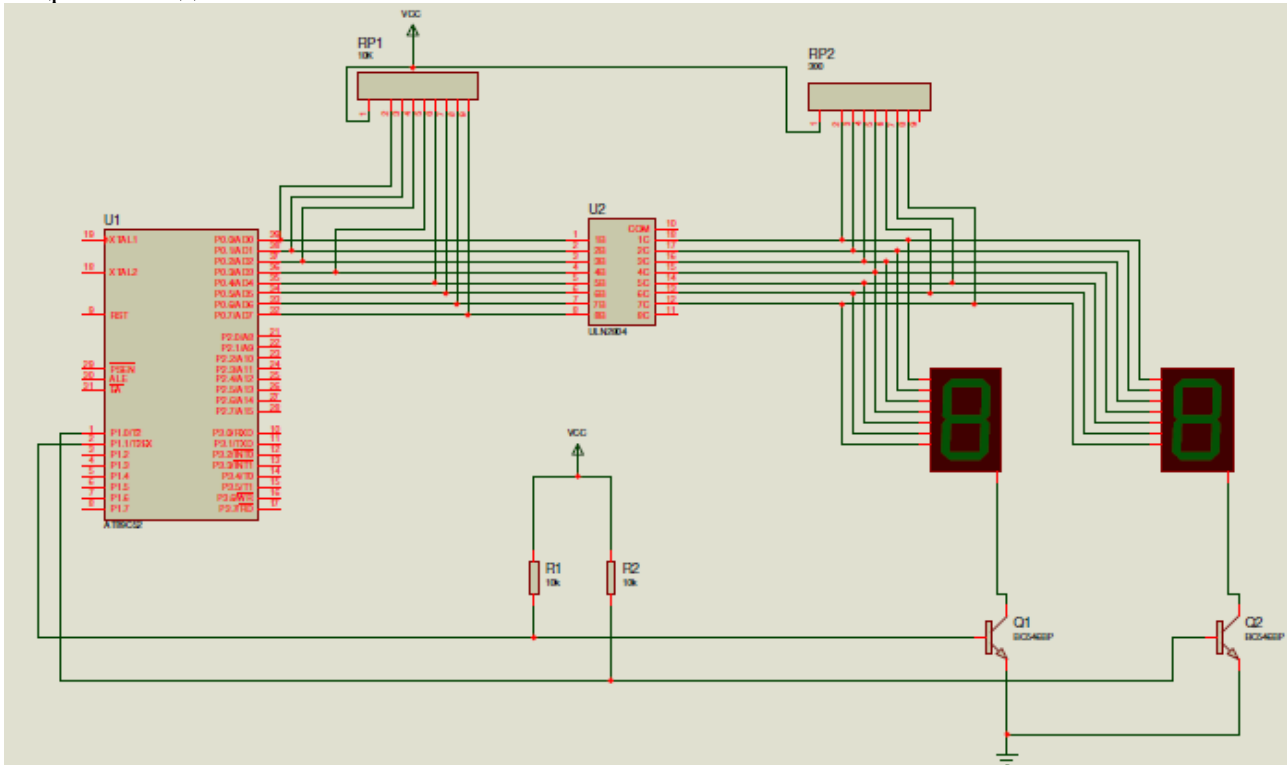
Мултиплексирање 7с дисплеја се често користи у микроконтролерским системима због уштеде у броју пинова који се користе за рад са њима (у овом режиму је неопходно свега 8 или 4 пина у случају BCD/7с декодера). Мултиплексирање се врши на начин да се великом брзином укључује час један, час други селекциони транзистор, при чему ове промјене, због тромости, око не може примијетити.

2. Кориштен софтвер и опрема

Кориштен је софтвер *MIDE-51* (за асемблирање) и софтверски пакет *Proteus* (за симулацију). За практичну верификацију кориштено је развојно окружење *Easy8051*.

3. Резултати и закључак

Кориштењем програма MIDE-51 и IV примјера генерисан је *lab_7.hex* фајл. У *Proteus*-у је нацртана следећа шема:



Сл. 7.2 Електрична шема система

Модификација система се састоји у пројектовању бројача модула 20, при чему би се стање бројача на пару 7с дисплеја.

Идеја се састојала у томе да референтни временски интервал у трајању 1s буде сигнализован путем одређеног броја прекида тајмера 0 (тачније њих 66). До тада се стање оба дисплеја мора ажурирати довољан број пута да се не примијети треперење. У прекидној рутини тајмера, по истеку 1s, инкрементује се тренутно стање бројача јединица, а када оно достигне број 9, инкрементује се и стање бројача десетица. Асемблерски код је дат у Листингу 7.1.

Листинг 7.1

```
DSEG
    org 20h
CSEG
    org 0000h
        jmp glavni
    org 0003h
        reti
    org 000bh
        jmp PREKID_T0
    org 0013h
        reti
    org 001bh
        reti
    org 0023h
        reti
    org 002bh
```



```

        reti
        org 0030h
glavni:
        mov r5,#0
        mov tmod , #01h ; tajmer T0 radi u modu 1
        mov ie , #82h
        mov th0 , #0D8h; ; u tajmer 0 D8F0
        mov tl0 , #0F0h ; sto odgovara prekidu tajmera svakih 15ms na 8MHz
        setb tr0 ; startovanje tajmera 0
POCETAK:
        mov dptr , #SEDMOSEG
        multipleksiraj:
        mov r0,#0
        mov r4,#0
ispis1:
        mov p1,#00h ;iskljucivanje oba tranzistora
        mov a,r0
        cjne a,#10,nastavi1
        mov r0,#0
        cjne r4,#10,dalje
        jmp multipleksiraj
dalje:
        mov a,r0
        nastavi1:
        movc a , @a+dptr ;ispis jedinica
        mov p0,a
        setb p1.0
        call kasnjenje
        clr p1.0
        mov a,r4 ;ispis desetica
        movc a , @a+dptr
        mov p0,a
        setb p1.1
        call kasnjenje
        jmp ispis1
SEDMOSEG:
        DB 0C0h,0F9h,0A4h,0B0h,99h,92h,82h,0F8h,80h,90h
;3Fh,06h,5Bh,4Fh,66h,6Dh,7Dh,07h,7Fh,6Fh ovo su konstante za A verziju razvojnog okruzenja
KASNJENJE:
        mov r3 , #5
KASNJENJE3:
        mov r2 , #10
KASNJENJE2:
        mov r1 , #50
KASNJENJE1:
        djnz r1 , KASNJENJE1
        djnz r2 , KASNJENJE2
        djnz r3 , KASNJENJE3
        ret
PREKID_T0:
        clr tr0 ; zaustavljamo tajmer 0
        inc r5
        cjne r5 , #66, PREKID_T0_1 ; poredimo r0 sa 66 i ako nije jednako znaci da nije
; istekla jedna sekunda jer je 66*15ms=1s
        mov r5 , #00h
    
```

```
inc r0 ; uvecavamo stanje jedinica
cjne r0,#10,PREKID_T0_1
inc r4 ; uvecavamo stanje desetica
cjne r4,#2,PREKID_T0_1
mov r4,#0
PREKID_T0_1:
mov th0 , #0D8h; ; u tajmer 0 D8F0
mov tl0 , #0F0h ; sto odgovara prekidu tajmera svakih 15ms na
8MHz
setb tr0 ; startamo tajmer 0
RETI
END
```

Вјежба бр. 8

LCD и RS232 СЕРИЈСКА КОМУНИКАЦИЈА

1. Кратак опис вјежбе

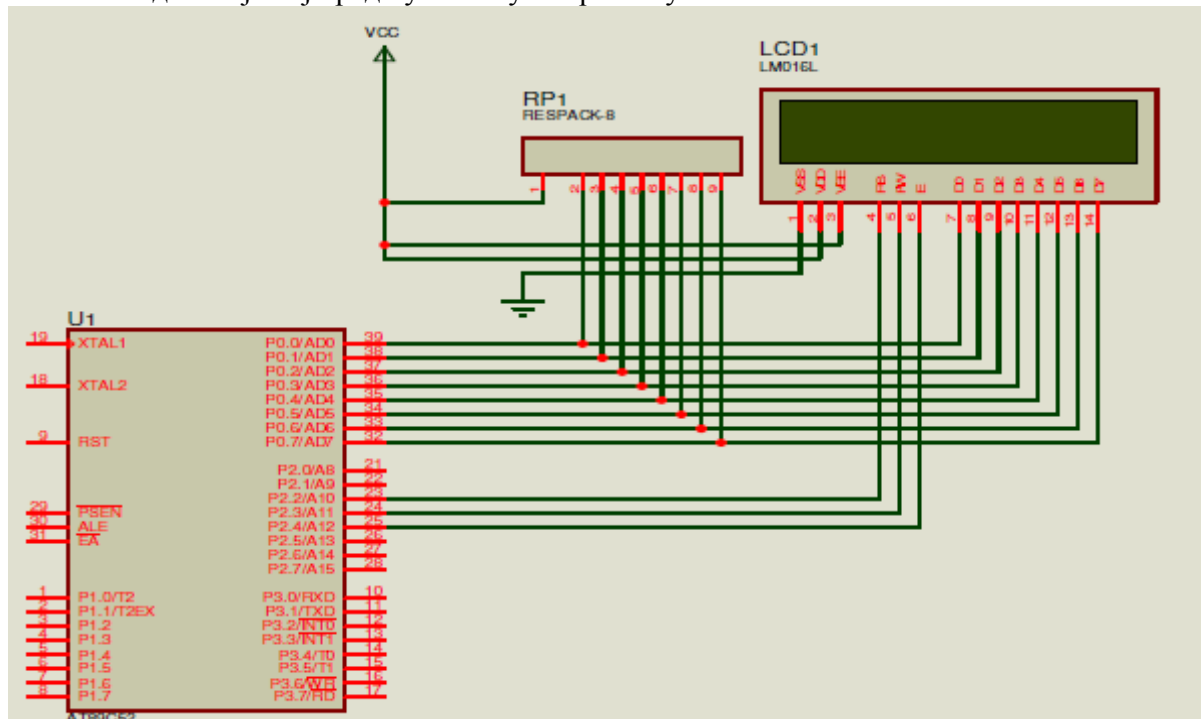
У овој вјежби реализована је серијска комуникација микроконтролера са рачунаром путем интерфејса за серијску комуникацију, а за приказ примљених података користио се LCD дисплеј. Као модификација захтијевала се промјена битске брзине уз додатне захтјеве за формом исписа на дисплеју.

2. Кориштен софтвер и опрема

Кориштен је софтвер *MIDE-51* (за асемблирање) и софтверски пакет *Proteus* (за симулацију). За практичну верификацију кориштено је развојно окружење *Easy8051*.

3. Резултати и закључак

У овој вјежби LCD ради у 8-битном режиму тако да укупно резервише 11 пинова микроконтролера. На слици 8.1 приказана је шема микроконтролерског система који се састоји од *Smart LCD* дисплеја који ради у поменутом режиму.



Сл. 8.1 Шема микроконтролерског система који се састоји од *Smart LCD* дисплеја који је спојен за рад у 8-битном режиму

Секција *pull-up* отпорника служи за дефинисање логичке 1 на порту 0. Три пина порта 0 служе као контролни пинови за сам дисплеј. Наиме, комуникација са дисплејем је једноставна и састоји се од слања контролних ријечи одговарајућег формата. Контролна ријеч се састоји од 11 бита. У листингу бр. 8.1 дат је асемблерски код који врши иницијализацију и испис текста на дисплеју.

Листинг 8.1

```

DSEG
    org 20h
CSEG
    org 0000h
        jmp glavni
    org 0003h
        reti
    org 000bh
        jmp PREKID_T0
    org 0013h
        reti
    org 001bh
        reti
    org 001bh
        reti
    org 0023h
        reti
    org 002bh
        reti

POCETAK:
org 0030h
    setb E ; enable postavljamo na visok nivo
    mov r0,#80h ; pocetna adresa prvog reda LCD displeja
    mov r5,#0C0h ; pocetna adresa drugog reda LCD displeja
    call INICIJALIZACIJA ; pozivamo potprogram za inicijalizaciju LC
    mov dptr,#PRVI_RED ; u dptr postavljamo adresu teksta koji cemo ispisati u prvom redu
    mov a,#00h ; resetujemo akumulator - ofset tabele na 0

PONOVO2:
    mov r4,a ; registar r4 služi za cuvanje ofseta
    movc a,@a+dptr ; citamo prvi karakter iz tabele
    cjne a,#'$',PONOVO1 ; poredimo ga sa $ karakterom posto sa njim završavamo string
    jmp PONOVO3

PONOVO1:
    call ispis_prvi_red ; pozivamo potprogram kojim ispisujemo karakter iz akumulatora na
    prvi red LCD-a
    mov a,r4
    inc a ; inkrementiramo ofset
    jmp PONOVO2 ; ovo ponavljamo sve dok ne iscitamo posljednji karakter

PONOVO3:
    mov dptr,#DRUGI_RED ; u dptr postavljamo adresu teksta koji cemo ispisati u drugom redu
    mov a,#00h ; resetujemo akumulator - ofset tabele1 na 0

PONOVO6:
    mov r4,a ; registar r4 služi za cuvanje ofseta
    movc a,@a+dptr ; citamo prvi karakter iz tabele1
    cjne a,#'$',PONOVO5 ; poredimo ga sa $ karakterom posto sa njim završavamo string
    jmp PONOVO

PONOVO5:

```

```
call ispis_drugi_red ; pozivamo potprogram kojim ispisujemo karakter iz akumulatora na
drugi red LCD
mov a,r4
```

```
PRVI_RED: DB "ZDRAVO!!!"
```

```
DRUGI_RED: DB "PDS", '$'
```

INICIJALIZACIJA:

```
setb E
clr RW
clr RS
mov p0,#01h ; brisemo sadrzaj displeja
clr E
call kasnjenje
setb E
mov p0,#00111000b ; podesavamo displej da radi kao dvoredni u 8 bitnom modu
clr E
call kasnjenje
setb E
mov p0,#0Fh ; ukljucujemo displej i kursor
clr E
call kasnjenje
setb E
RET
```

ISPIS_PRVI_RED:

```
mov p0,r0 ;adresa karaktera
inc r0
clr RW
clr RS
clr E
call kasnjenje
setb E
mov p0,a
setb RS
clr E
call kasnjenje
setb E
RET
```

ISPIS_DRUGI_RED:

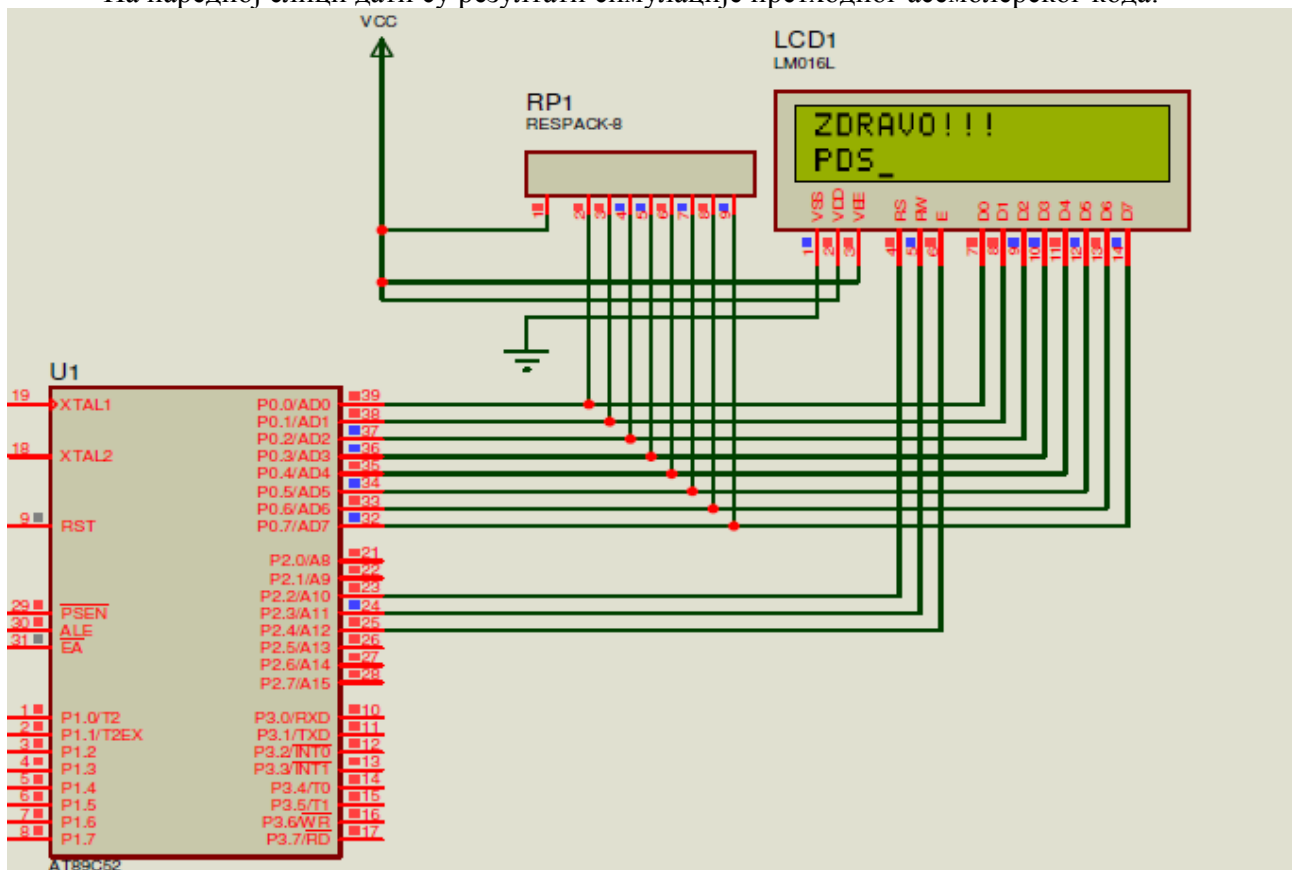
```
mov p0,r5
inc r5
clr RW
clr RS
clr E
call kasnjenje
setb E
mov p0,a
setb RS
clr E
call kasnjenje
setb E
```

```

ret
RET

KASNJENJE:
    mov r3 , #1
KASNJENJE3:
    mov r2 , #100
KASNJENJE2:
    mov r1 , #255
KASNJENJE1:
    djnz r1 , KASNJENJE1
    djnz r2 , KASNJENJE2
    djnz r3 , KASNJENJE3
    ret
END
    
```

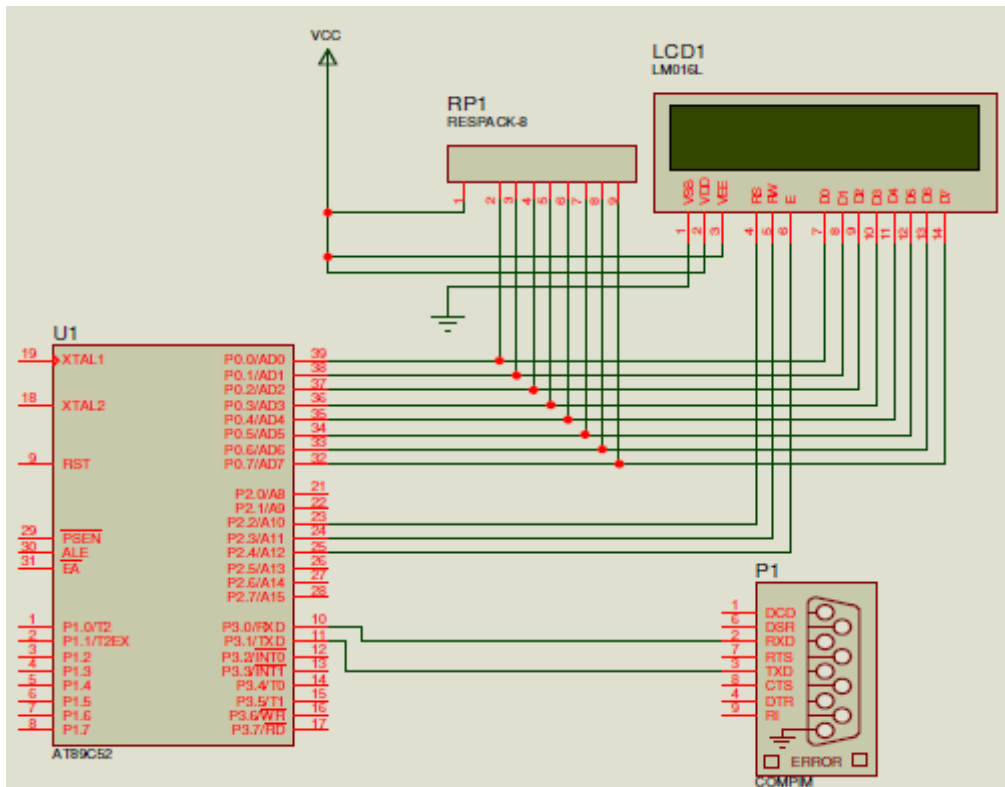
На наредној слици дати су резултати симулације претходног асемблерског кода:



Сл. 8.2 Резултати симулације

Приликом комуникације са дисплејем неопходно је обезбиједити софтверско кашњење.

На слици 8.3 дата је електрична шема система микроконтролерског система Smart LCD дисплеја који је спојен за рад у 8-битном режиму и серијског интерфејса реализованог са компонентом COMPIM. Асемблерски код који је дат у припреми реализује слање карактера *a* рачунару битском брзином од 1200 бита по секунди и пријем карактера са рачунара који се потом исписује на дисплеју.



Сл. 8.3 Шема микроконтролерског система који се састоји од Smart LCD дисплеја који је спојен за рад у 8-битном режиму и серијског интерфејса

Кориштењем програма за креирање виртуелних портова на рачунару потребно је креирати пар портова који су међусобно повезани. Један од њих треба подесити као излазни у компоненти COMPIM, а са другим портом остварити везу преко клијентске конзоле за серијску комуникацију.

С' обзиром да се као генератор *baudrate-a* користи тајмер T0 који ради у *auto-reload* режиму и чињеницом да је бројач иницијализован на вриједност 239, на основу релације:

$$BRZINA = \frac{2^{SMOD} * f_{osc}}{12 \cdot 32 \cdot (256 - TH1)}$$

слиједи да је фреквенција осцилатора контролера 8MHz.

Када податак буде примљен долази до прекида од стране пријемника серијске комуникације који се манифестује сетовањем одговарајућег бита у SCON регистру. Примљени податак се потом шаље на дисплеј. Претходне операције извршава следећи дио кода (процедура испис је слична претходним процедурама за испис првог односно другог реда):

Листинг 8.2

SERIJSKA:

```
jb ri, PRIJEM ; ako je ri setovan radi se o prijemu u suprotnom o predaji podatka
clr ti ; mora se softverski obrisati
RETI ; vraćanje iz potprograma
```

PRIJEM:

```
mov a,sbuf ; procedura prijema-primljeni karakter u akumulator
call ispis ; ispisujemo karakter iz akumulatora na LCD
clr ri ; mora se softverski obrisati
RETI ; vraćanje iz potprograma
```

ISPIS:

```
mov p0,r0 ; adresa
inc r0
clr RW
```



```
clr RS
clr E
call kasnjenje
setb E

mov p0,a ;karakter
setb RS
clr E
call kasnjenje
setb E
```

Модификација се састојала у промјени симболске брзине и реализовања ефекта еха, тј. карактерв који је послан од стране рачунара треба да буде враћен рачунару.

Корекција брзине је извршена промјеном садржаја тајмера који је сада износи 253. Затим, одмах по пријему симбола извршено је његово слање рачунару.

Листинг 8.3

PRIJEM:

```
mov a,sbuf ; procedura prijema-primljeni karakter u akumulator
call ispis ; ispisujemo karakter iz akumulatora na LCD
clr ri ; mora se softverski obrisati
mov sbuf,a ;eho
RETI ; vraćanje iz potprograma
```

Потом је модификована процедура за испис због потребе да се симболи исписују у два реда када се за то јави потреба:

Листинг 8.4

ISPIS:

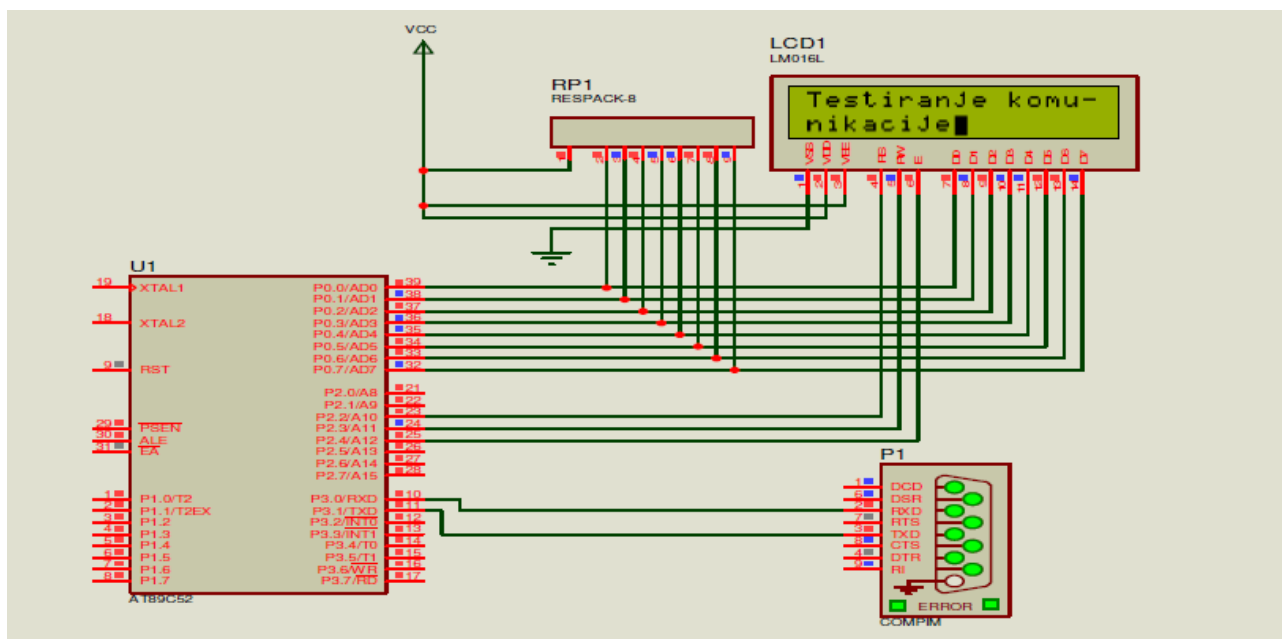
```
inc r6 ;r6 je inicijalizovana na pocetku programa sa 0
cjne r6,#17,postavi ;kraj prvog reda
mov r5,#1 ;kontrolni registar
jmp proceed
postavi:
mov r5,#0
```

proceed:

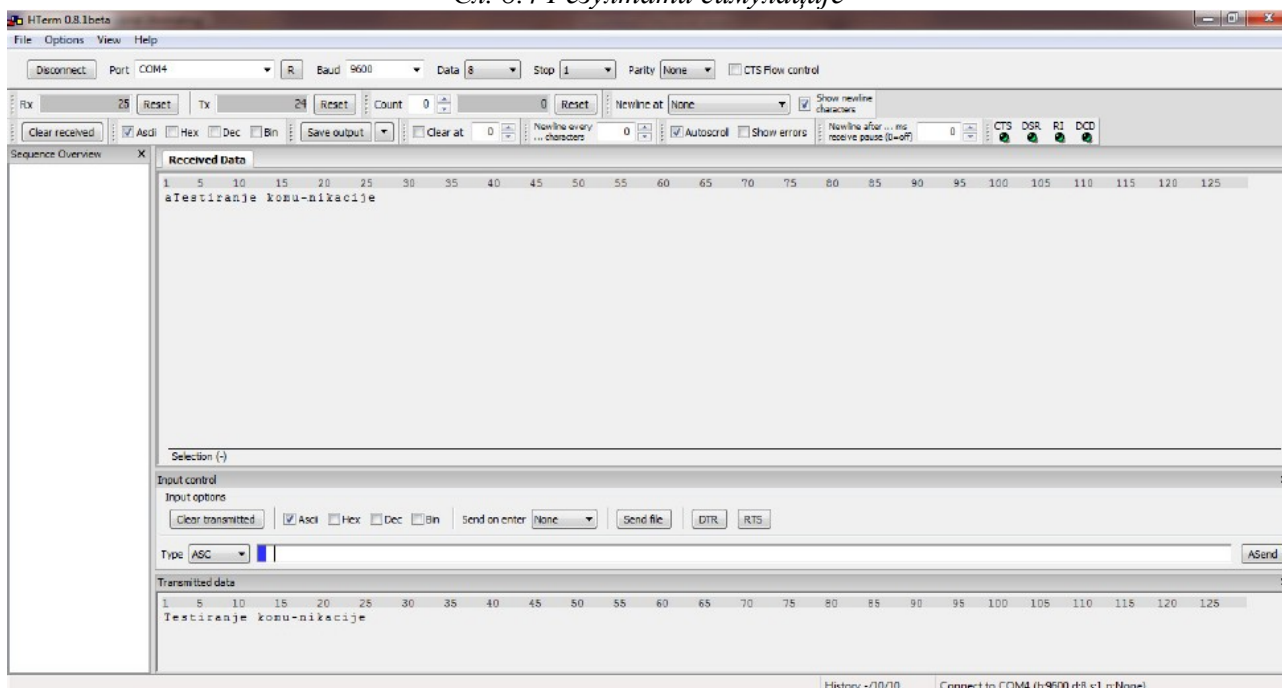
```
cjne r5,#0,nastavi
mov p0,r0 ;ispis u drugom redu
inc r0
clr RW
clr RS
clr E
call kasnjenje
setb E
jmp kraj
```

nastavi:

```
dec r6 ;ispis u prvom redu
mov p0,r7
inc r7
clr RW
clr RS
clr E
call kasnjenje
setb E
```



Сл. 8.4 Резултати симулације



Сл. 8.5 Hterm терминал