Cvičenie 6
Konečné automaty

Zadanie:

Realizujte stopky, ktoré budú odpočítavať čas z dvojcifernej preddefinovanej hodnoty po nulu. Pri nule nech odznie prerušovaný zvukový signál. Počiatočná hodnota nech je preddefinovaná len na mieste desiatok pomocou dip-swichov maximálnym číslom 7 (kvôli obmedzený zdrojom CPLD). Odpočítavanie má byť spustené mikro-tlačidlom. Reset má byť tiež vnútený mikro-tlačidlom. Prvý dip-swich použite na prepínanie módu pre nastavenie počiatočnej hodnoty. Zvyšné dip-swiche použite na nastavenie počiatočnej hodnoty. Tieto dip-switche majú byť načítané len v režime nastavenia počiatočnej hodnoty. Po odpočítaní požadovaného času a odznení zvukového signálu sa majú stopky nastaviť na počiatočnú hodnotu a čakať na stlačenie tlačidla pre štart alebo sa môžu prepnúť do režimu nastavenia počiatočnej hodnoty. Napíšte Testbench súbor a projekt odsimulujte a takisto syntetizujte v HW.

Riešenie:

Výklad k celému projektu bude uskutočnený na cvičení.

STATE_DEMO.vhd   (top-entity)

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

entity STATE_DEMO is
port(
    CLK50MHZ    : in std_logic;
    RESET       : in std_logic;
    DISPLAY1    : out std_logic_vector(7 downto 0);
    DISPLAY10   : out std_logic_vector(7 downto 0);
    START_BUT   : in std_logic;
    SET_BUT     : in std_logic;
    SET_SWICH   : in std_logic_vector(2 downto 0);
    BUZZER_PLUS : out std_logic;
    BUZZER_MINUS: out std_logic

    );
end STATE_DEMO;

architecture arch of STATE_DEMO is

  constant DIV_FACTOR_SM   : natural range 100000 to 25000000 :=  2500000;
  constant DIV_FACTOR_BUZ  : natural range 1000   to 100000   :=  50000;
  constant LIMIT           : natural range 1 to 15         := 12;
  constant INCREMENT       : natural range 1 to 5          := 2;

  signal clk_sm         : std_logic;
  signal clk_buzzer     : std_logic;
  signal bcdcnt1        : std_logic_vector(3 downto 0);
  signal bcdcnt10       : std_logic_vector(2 downto 0);
  signal buzzer_enable  : std_logic;

  component PRESCALER is
    generic(
    DIV_FACTOR_SM   : natural range 100000 to 25000000 :=  10;
    DIV_FACTOR_BUZ  : natural range 1000   to 100000   :=  10
    );
    port(
      CLK50MHZ    : in std_logic;
      RESET       : in std_logic;
      CLKOUT1     : out std_logic;
      CLKOUT2     : out std_logic
    );
  end component;
  component SM is
    port(
      CLK         : in std_logic;
      RESET       : in std_logic;
      CNTOUT1     : out std_logic_vector(3 downto 0);
      CNTOUT10    : out std_logic_vector(2 downto 0);
      START_BUT   : in std_logic;
      SET_BUT     : in std_logic;
      SET_SWICH   : in std_logic_vector(2 downto 0);
      BUZZER_EN   : out std_logic
    );
  end component;

  component BCD2SEG is
    port(
      BCDIN       : in std_logic_vector  (3 downto 0);
      DISPLAY     : out std_logic_vector (7 downto 0)
    );
  end component;

  component BUZZER is
    port(
      CLK_DRIVE   : in std_logic;
      ENABLE      : in std_logic;
      OUT_PLUS    : out std_logic;
      OUT_MINUS   : out std_logic
    );
  end component;

begin

  the_PRESCALER: PRESCALER
    generic map(
      DIV_FACTOR_SM  => DIV_FACTOR_SM,
      DIV_FACTOR_BUZ => DIV_FACTOR_BUZ
    )
    port map(
      CLK50MHZ   => CLK50MHZ,
      RESET      => RESET,
      CLKOUT1    => clk_buzzer,
      CLKOUT2    => clk_sm
    );

  the_SM: SM
    port map(
      CLK        => clk_sm,
      RESET      => RESET,
      CNTOUT1    => bcdcnt1,
      CNTOUT10   => bcdcnt10,
      START_BUT  => START_BUT,
      SET_BUT    => SET_BUT,
      SET_SWICH  => SET_SWICH,
      BUZZER_EN  => buzzer_enable
    );

  the_BCD2SEG1: BCD2SEG
    port map(
      BCDIN      => bcdcnt1,
      DISPLAY    => DISPLAY1
    );

  the_BCD2SEG10: BCD2SEG
    port map(
      BCDIN      => '0' & bcdcnt10,
      DISPLAY    => DISPLAY10
    );

   the_BUZZER: BUZZER
    port map(
      CLK_DRIVE  => clk_buzzer,
      ENABLE     => buzzer_enable,
      OUT_PLUS   => BUZZER_PLUS,
      OUT_MINUS  => BUZZER_MINUS
    );

end arch;
```

SM.vhd

```
-- konecny automat
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all; -- using comparison ""<""

entity SM is
  port(
    CLK         : in std_logic;
    RESET       : in std_logic;
    CNTOUT1     : out std_logic_vector(3 downto 0);
    CNTOUT10    : out std_logic_vector(2 downto 0);
    START_BUT   : in std_logic;
    SET_BUT     : in std_logic;
    SET_SWICH   : in std_logic_vector(2 downto 0);
    BUZZER_EN   : out std_logic
  );
end SM;

architecture arch of SM is

  type my_state is (wait_for_start, refresh, run_down, set_time, finish);

  signal cnt1         : std_logic_vector (3 downto 0); -- citac jednotiek
  signal cnt10        : std_logic_vector (2 downto 0); -- citac desiatok
  signal cnt10_preset : std_logic_vector (2 downto 0); -- prednastavenie desiatok
  signal cnt_finish   : std_logic_vector (3 downto 0); -- citac, ktory zabezpecuje
prerusovany ton
  signal state        : my_state;                      -- konecny automat

begin

  CNTOUT1 <=  cnt1;
  CNTOUT10 <= cnt10;
  BUZZER_EN <= '1' when ((state = finish) and (cnt_finish(0) = '1')) else '0';

  process(RESET,CLK)
    begin
      if (RESET = '0') then
         cnt1 <= "0000";
         cnt10 <= "111";
         cnt10_preset <= "111";
         state <= wait_for_start;
         cnt_finish <= (others => '1');
      elsif (rising_edge(CLK)) then

         case state is
           when wait_for_start =>         -- cakanie na start odpocitavania alebo
             if (SET_BUT = '0') then      -- nastavenie pociatocnej hodnoty
               state <= set_time;
             elsif (START_BUT = '0') then
               state <= run_down;
             else
               state <= wait_for_start;
             end if;

           when run_down =>               -- zaciatok odpocitavania
             if (cnt1 /= 0) then
               cnt1 <= cnt1 - 1;
             else
               cnt1 <= "1001";
               if(cnt10 /= 0) then
                 cnt10 <= cnt10 - 1;
               else
                 cnt10 <= "000";
                 cnt1  <= "0000";
                 state <= finish;
               end if;
             end if;

           when finish =>                 -- generovanie prerusovaneho tonu
             cnt10 <= "000";
             cnt1  <= "0000";
             if (cnt_finish /= 0) then
               cnt_finish <= cnt_finish - 1;
             else
               state <= refresh;
             end if;

           when refresh =>                -- nacitanie pociatocnej hodnoty
             cnt10 <= cnt10_preset;
             cnt1  <= "0000";
             state <= wait_for_start;
             cnt_finish <= (others => '1');

           when set_time =>               -- nastavenie pociatocnej hodnoty
             cnt10 <= not SET_SWICH;
             cnt10_preset <= not SET_SWICH;
             cnt1  <= "0000";
             if (SET_BUT = '1') then
               state <= wait_for_start;
             end if;

         end case;

      end if;
    end process;
end arch;
```

PRESCALER.vhd

```
-- prescaler
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all; -- using comparison ""<""

entity PRESCALER is
  generic(
    DIV_FACTOR_SM   : natural range 100000 to 25000000 :=  10;
    DIV_FACTOR_BUZ  : natural range 1000   to 100000   :=  10
  );
  port(
    CLK50MHZ    : in std_logic;
    RESET       : in std_logic;
    CLKOUT1     : out std_logic;
    CLKOUT2     : out std_logic
  );
end PRESCALER;

architecture arch of PRESCALER is

    signal cnt1  : std_logic_vector (24 downto 0);
    signal cnt2  : std_logic_vector (24 downto 0);
    signal clk1  : std_logic;
    signal clk2  : std_logic;

begin

  CLKOUT1 <= clk1;
  CLKOUT2 <= clk2;

  process(RESET,CLK50MHZ)
    begin
      if (RESET = '0') then
         cnt1 <= (others => '0');
         cnt2 <= (others => '0');
         clk1 <= '0';
         clk2 <= '0';
      elsif (rising_edge(CLK50MHZ)) then
         if (cnt1 < DIV_FACTOR_BUZ/2) then
           cnt1 <= cnt1 + 1;
         else
           cnt1 <= (others => '0');
           clk1 <= not clk1;
           if (cnt2 < DIV_FACTOR_SM/DIV_FACTOR_BUZ) then
             cnt2 <= cnt2 + 1;
           else
             cnt2 <= (others => '0');
             clk2 <= not clk2;
           end if;
         end if;
      end if;
    end process;
end arch;
```

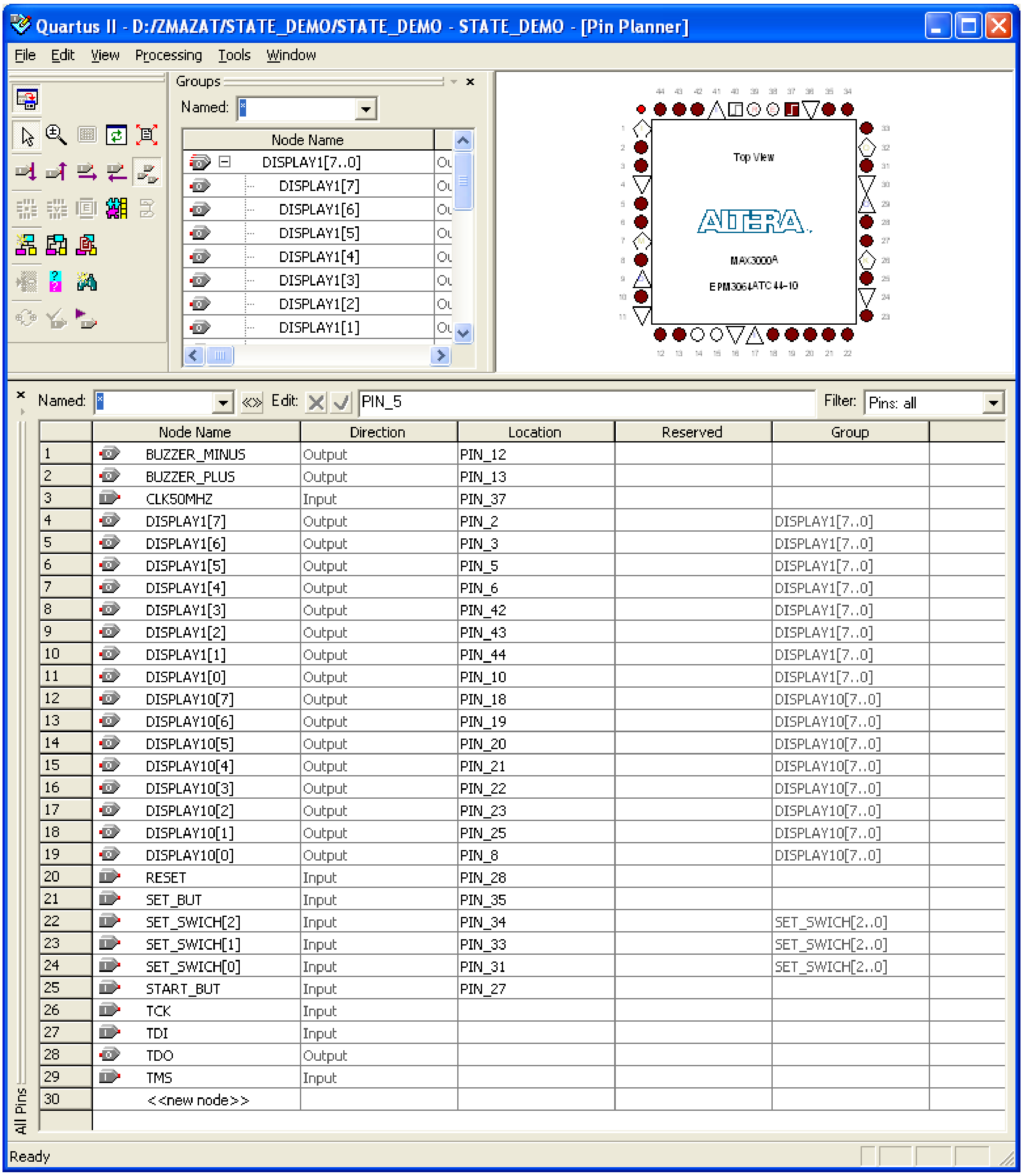
BCD2SEG.vhd

```
-- decoder
library ieee;
use ieee.std_logic_1164.all;

entity BCD2SEG is
  port(
    BCDIN   : in std_logic_vector  (3 downto 0);
    DISPLAY : out std_logic_vector (7 downto 0)
  );
end BCD2SEG;

architecture arch of BCD2SEG is
begin
  with BCDIN select
    DISPLAY <=
      "00000011" when "0000", -- 0
      "10011111" when "0001", -- 1
      "00100101" when "0010", -- 2
      "00001101" when "0011", -- 3
      "10011001" when "0100", -- 4
      "01001001" when "0101", -- 5
      "01000001" when "0110", -- 6
      "00011111" when "0111", -- 7
      "00000001" when "1000", -- 8
      "00001001" when "1001", -- 9
      "00010001" when "1010", -- A
      "11000001" when "1011", -- b
      "11100101" when "1100", -- c
      "10000101" when "1101", -- d
      "01100001" when "1110", -- E
      "01110001" when "1111", -- F
      "11111111" when others; -- pre ine moznosti
end arch;
```

BUZZER.vhd

```
-- buzzer
library ieee;
use ieee.std_logic_1164.all;

entity BUZZER is
  port(
    CLK_DRIVE   : in std_logic;
    ENABLE      : in std_logic;
    OUT_PLUS    : out std_logic;
    OUT_MINUS   : out std_logic
  );
end BUZZER;

architecture arch of BUZZER is

begin

   OUT_PLUS  <= CLK_DRIVE when (ENABLE = '1') else '0';
   OUT_MINUS <= not CLK_DRIVE when (ENABLE = '1') else '1';

end arch;
```

Priradenie pinov:


Quartus II - D:/ZMAZAT/STATE_DEMO/STATE_DEMO - STATE_DEMO - [Pin Planner]
File Edit View Processing Tools Window
Groups
Named:
Top View
ALTERA
MAX3000A
EPM3064ATC44-10
Named:
Edit:
Filter:
All Pins
Ready

Node Name	
DISPLAY1[7..0]	Ou
DISPLAY1[7]	Ou
DISPLAY1[6]	Ou
DISPLAY1[5]	Ou
DISPLAY1[4]	Ou
DISPLAY1[3]	Ou
DISPLAY1[2]	Ou
DISPLAY1[1]	Ou

Edit: PIN_5

Filter: Pins: all

	Node Name	Direction	Location	Reserved	Group
1	BUZZER_MINUS	Output	PIN_12		
2	BUZZER_PLUS	Output	PIN_13		
3	CLK50MHZ	Input	PIN_37		
4	DISPLAY1[7]	Output	PIN_2		DISPLAY1[7..0]
5	DISPLAY1[6]	Output	PIN_3		DISPLAY1[7..0]
6	DISPLAY1[5]	Output	PIN_5		DISPLAY1[7..0]
7	DISPLAY1[4]	Output	PIN_6		DISPLAY1[7..0]
8	DISPLAY1[3]	Output	PIN_42		DISPLAY1[7..0]
9	DISPLAY1[2]	Output	PIN_43		DISPLAY1[7..0]
10	DISPLAY1[1]	Output	PIN_44		DISPLAY1[7..0]
11	DISPLAY1[0]	Output	PIN_10		DISPLAY1[7..0]
12	DISPLAY10[7]	Output	PIN_18		DISPLAY10[7..0]
13	DISPLAY10[6]	Output	PIN_19		DISPLAY10[7..0]
14	DISPLAY10[5]	Output	PIN_20		DISPLAY10[7..0]
15	DISPLAY10[4]	Output	PIN_21		DISPLAY10[7..0]
16	DISPLAY10[3]	Output	PIN_22		DISPLAY10[7..0]
17	DISPLAY10[2]	Output	PIN_23		DISPLAY10[7..0]
18	DISPLAY10[1]	Output	PIN_25		DISPLAY10[7..0]
19	DISPLAY10[0]	Output	PIN_8		DISPLAY10[7..0]
20	RESET	Input	PIN_28		
21	SET_BUT	Input	PIN_35		
22	SET_SWICH[2]	Input	PIN_34		SET_SWICH[2..0]
23	SET_SWICH[1]	Input	PIN_33		SET_SWICH[2..0]
24	SET_SWICH[0]	Input	PIN_31		SET_SWICH[2..0]
25	START_BUT	Input	PIN_27		
26	TCK	Input			
27	TDI	Input			
28	TDO	Output			
29	TMS	Input			
30	<<new node>>				