Protokol Výtah

Zadání:

Budova má 7 pater a přízemí. Volbu patra simulujte pomocí sw0-sw7. Reset realizujte pomocí btn0. Na sedmisegmentu zobrazte, v kterém patře se nacházíte. Pomocí Led0-Led2 zobrazte počet pater o který se posunete v binárním kódu např. o 6. pater 110 a pomocí Led4 posun dolů a Led5 posun nahoru. Pokud použijete více sw najednou rozsvítí se Led0-Led5. Pokud nepoužijete žádný sw, Led jsou zhasnuty a zůstáváte ve stávajícím patře.

Teoretický rozbor Spartan a VHDL:

Spartan je programovatelné hradlové pole, což je typ logického integrovaného obvodu, který je vyroben tak, aby mohl být naprogramován kdykoliv. Obsahuje pole programovatelných logických obvodů (PLD), logických bloků, umožňuje je navzájem propojit a tím vytvořit takřka libovolné číslicové zařízení. Mikroprocesor je víceúčelové programovatelné zařízení, které na vstupu akceptuje digitální data, zpracuje je pomocí instrukcí uložených v paměti a jako výstup zobrazí výsledek. Mikroprocesor představuje příklad sekvenčního logického obvodu, který pro uložení dat používá dvojkovou soustavu.

VHDL je programovací jazyk, který slouží pro popis hardwaru. Používá se pro návrh a simulaci digitálních integrovaných obvodů, například programovatelných hradlových polí nebo různých zákaznických obvodů. Umožňuje návrh jak logických tak i sekvenčních struktur a jeho hlavní výhodou je jeho univerzálnost.

Definice stavů a jejich kódování:

Vstupní proměnné:

	sw7	sw6	sw5	sw4	sw3	sw2	sw1	sw0
X0	0	0	0	0	0	0	0	0
X1	0	0	0	0	0	0	0	1
X2	0	0	0	0	0	0	1	0
X3	0	0	0	0	0	1	0	0
X4	0	0	0	0	1	0	0	0
X5	0	0	0	1	0	0	0	0
X6	0	0	1	0	0	0	0	0
X7	0	1	0	0	0	0	0	0
x8	1	0	0	0	0	0	0	0

Vnitřní proměnné:

	Q2	Q1	Q0
S0	0	0	0
S1	0	0	1
S2	0	1	0
S3	0	1	1
S4	1	0	0
S5	1	0	1
S6	1	1	0
S7	1	1	1

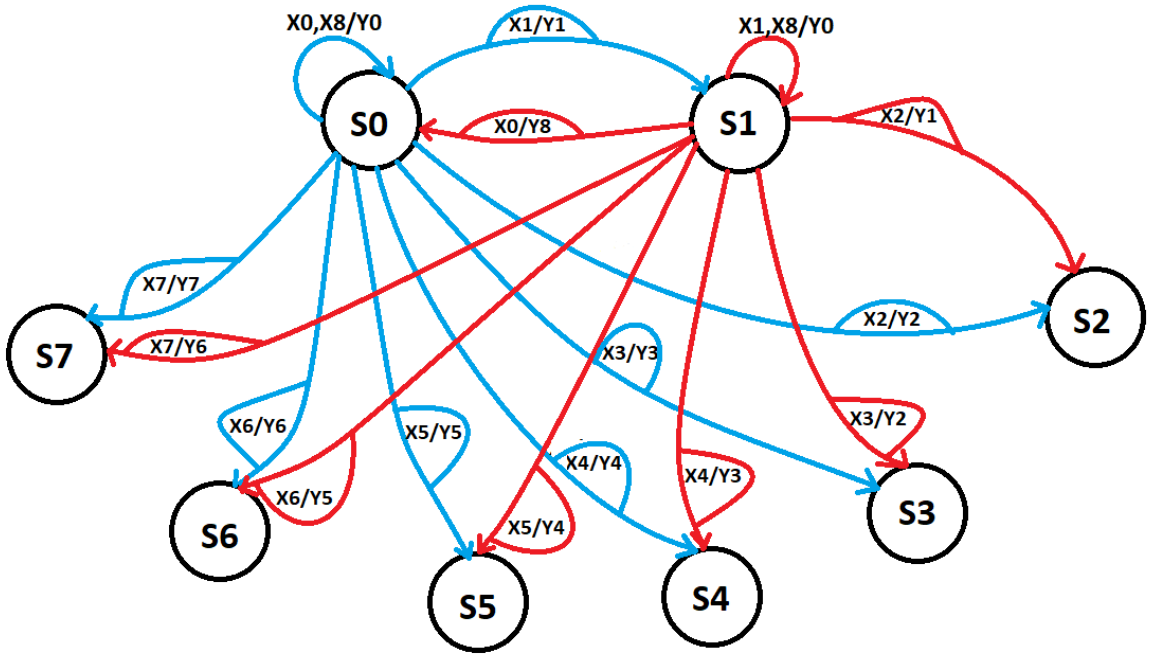
Výstupní proměnné:

	L5	L4	L3	L2	L1	L0
Y0	0	0	0	0	0	0
Y1	1	0	0	0	0	1
Y2	1	0	0	0	1	0
Y3	1	0	0	0	1	1
Y4	1	0	0	1	0	0
Y5	1	0	0	1	0	1
Y6	1	0	0	1	1	0
Y7	1	0	0	1	1	1
Y8	0	1	0	0	0	1
Y9	0	1	0	0	1	0
Y10	0	1	0	0	1	1
Y11	0	1	0	1	0	0
Y12	0	1	0	1	0	1
Y13	0	1	0	1	1	0
Y14	0	1	0	1	1	1

Popis automatu Mealy:

Mealyho automat se označuje jako konečný automat s výstupem. Výstup je generován na základě příchozího vstupu i momentálního stavu. Jeho stavový diagram automatu má ke každému přechodu přiřazenu nejen vstupní hodnotu, kterou je přechod aktivován, ale i výstupní hodnotu, která je při aktivaci přechodu vygenerována.

Orientovaný graf:



Tabulky přechodů mezi vnitřními stavy v závislosti na vstupních stavech, tabulky výstupů:

	X0/Y	X1/Y	X2/Y	X3/Y	X4/Y	X5/Y	X6/Y	X7/Y
s0	S0,S0/Y0,Y0	S1,S0/Y1,Y8	S2,S0/Y2,Y9	S3,S0/Y3,Y0	S4,S0/Y4,Y0	S5,S0/Y5,Y0	S6,S0/Y6,Y0	S7,S0/Y7,Y0
s1	S1,S1/Y0,Y0	S2,S0/Y1,Y8	S3,S0/Y2,Y8	S4,S0/Y3,Y8	S5,S0/Y4,Y8	S6,S0/Y5,Y8	S7,S0/Y6,Y8	S7,S0/Y6,Y8
s2	S2,S2/Y0,Y0	S3,S1/Y1,Y8	S4,S0/Y2,Y9	S5,S0/Y3,Y9	S6,S0/Y4,Y9	S7,S0/Y5,Y9	S7,S0/Y5,Y9	S7,S0/Y5,Y9
s3	S3,S3/Y0,Y0	S4,S2/Y1,Y8	S5,S1/Y2,Y9	S6,S0/Y3,Y10	S7,S0/Y4,Y10	S7,S0/Y4,Y10	S7,S0/Y4,Y10	S7,S0/Y4,Y10
s4	S4,S4/Y0,Y0	S5,S3/Y1,Y8	S6,S2/Y2,Y9	S7,S1/Y3,Y10	S7,S0/Y3,Y10	S7,S0/Y3,Y11	S7,S0/Y3,Y11	S7,S0/Y3,Y11
s5	S5,S5/Y0,Y0	S6,S4/Y1,Y8	S7,S2/Y2,Y9	S7,S2/Y2,Y10	S7,S1/Y2,Y11	S7,S0/Y2,Y12	S7,S0/Y2,Y12	S7,S0/Y2,Y12
s6	S6,S6/Y0,Y0	S7,S5/Y1,Y8	S6,S4/Y0,Y9	S7,S3/Y1,Y10	S7,S2/Y1,Y11	S7,S1/Y1,Y12	S7,S0/Y1,Y13	S7,S0/Y1,Y13
s7	S7,S7/Y0,Y0	S8,S6/Y1,Y8	S7,S5/Y0,Y9	S7,S4/Y0,Y10	S7,S3/Y0,Y11	S7,S2/Y0,Y12	S7,S1/Y0,Y13	S7,S0/Y0,Y14

Dělička:

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

entity delicka is
    Port ( CLK_in : in  STD_LOGIC;
           CLK_out : out  STD_LOGIC);
end delicka;

architecture Behavioral of delicka is

begin

    process (CLK_in)
        variable i : integer range 0 to 15000000 ;
begin
    if rising_edge(CLK_in) then
        if i=0 then CLK_out <= '1' ;
                    i := 9843000 ;
        else
                CLK_out <= '0' ;
                i := i - 1 ;
         end if ;
     end if ;
    end process;

end Behavioral;
```

Dekodér:

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



entity dekoder is
    Port ( HEX : in  STD_LOGIC_VECTOR (2 downto 0);
           LED : out  STD_LOGIC_VECTOR (6 downto 0)
              );
end dekoder;


architecture Behavioral of dekoder is

begin


    with HEX SELect
   LED<= "1111001" when "001",   --1
         "0100100" when "010",   --2
         "0110000" when "011",   --3
         "0011001" when "100",   --4
         "0100100" when "101",   --5
         "0100000" when "110",   --6
         "0001111" when "111",   --7
         "1000000" when others;   --0

end Behavioral;
```

Výtah:

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3  use IEEE.NUMERIC_STD.ALL;
4  entity vytahl is
5
6   Port ( patro : inout STD_LOGIC_VECTOR (2 downto 0);
7         clk : in STD_LOGIC;
8         reset : in STD_LOGIC;
9         kam : in STD_LOGIC_VECTOR (7 downto 0);
10        vyst : out STD_LOGIC_VECTOR (5 downto 0));
11 end vytahl;
12
13 architecture Behavioral of vytahl is
14  signal state, next_state : STD_LOGIC_VECTOR (2 downto 0);
15
16
17    constant s0 : STD_LOGIC_VECTOR (2 downto 0) := "000";
18    constant s1 : STD_LOGIC_VECTOR (2 downto 0) := "001";
19    constant s2 : STD_LOGIC_VECTOR (2 downto 0) := "010";
20    constant s3 : STD_LOGIC_VECTOR (2 downto 0) := "011";
21    constant s4 : STD_LOGIC_VECTOR (2 downto 0) := "100";
22    constant s5 : STD_LOGIC_VECTOR (2 downto 0) := "101";
23    constant s6 : STD_LOGIC_VECTOR (2 downto 0) := "110";
24    constant s7 : STD_LOGIC_VECTOR (2 downto 0) := "111";
25
26
27 begin
28    SYNC_PROC: process (clk)
29          begin
30          if rising_edge (clk)
31             then if (reset='0')
32                then state <= next_state;
33                else state <= s0;
34             end if;
35          end if;
36  end process SYNC_PROC;
```

```
38  OUTPUT_DECODE: process (state, kam)
39 Begin
40 case (state) is
41
42       when s0 =>
43             if (kam = "00000001") then vyst <= "000000";       --0
44             elsif (kam = "00000010") then vyst <= "100001";    --1
45             elsif (kam = "00000100") then vyst <= "100010";    --2
46             elsif (kam = "00001000") then vyst <= "100011";    --3
47             elsif (kam = "00010000") then vyst <= "100100";    --4
48             elsif (kam = "00100000") then vyst <= "100101";    --5
49             elsif (kam = "01000000") then vyst <= "100110";    --6
50             elsif (kam = "10000000") then vyst <= "100111";    --7
51             elsif (kam = "00000000") then vyst <= "000000";    --00
52
53
54          else vyst <= "111111";
55       end if;
56
57       when s1 =>
58
59             if (kam = "00000001") then vyst <= "010001";       --0
60             elsif (kam = "00000010") then vyst <= "000000";    --1
61             elsif (kam = "00000100") then vyst <= "100001";    --2
62             elsif (kam = "00001000") then vyst <= "100010";    --3
63             elsif (kam = "00010000") then vyst <= "100011";    --4
64             elsif (kam = "00100000") then vyst <= "100100";    --5
65             elsif (kam = "01000000") then vyst <= "100101";    --6
66             elsif (kam = "10000000") then vyst <= "100110";    --7
67             elsif (kam = "00000000") then vyst <= "000000";    --00
68
69          else vyst <= "111111";
70       end if;
71
72       when s2 =>
73
74             if (kam = "00000001") then vyst <= "010010";       --0
75             elsif (kam = "00000010") then vyst <= "010010";    --1
76             elsif (kam = "00000100") then vyst <= "000000";    --2
77             elsif (kam = "00001000") then vyst <= "100001";    --3
78             elsif (kam = "00010000") then vyst <= "100010";    --4
79             elsif (kam = "00100000") then vyst <= "100011";    --5
80             elsif (kam = "01000000") then vyst <= "100100";    --6
81             elsif (kam = "10000000") then vyst <= "100101";    --7
82             elsif (kam = "00000000") then vyst <= "000000";    --00
83
84          else vyst <= "111111";
85       end if;
```

```
      when s3 =>

            if (kam = "00000001") then vyst <= "010011";       --0
            elsif (kam = "00000010") then vyst <= "010010";    --1
            elsif (kam = "00000100") then vyst <= "010001";    --2
            elsif (kam = "00001000") then vyst <= "000000";    --3
            elsif (kam = "00010000") then vyst <= "100001";    --4
            elsif (kam = "00100000") then vyst <= "100010";    --5
            elsif (kam = "01000000") then vyst <= "100011";    --6
            elsif (kam = "10000000") then vyst <= "100100";    --7
            elsif (kam = "00000000") then vyst <= "000000";    --00

         else vyst <= "111111";
      end if;


      when s4 =>

            if (kam = "00000001") then vyst <= "010100";       --0
            elsif (kam = "00000010") then vyst <= "010011";    --1
            elsif (kam = "00000100") then vyst <= "010010";    --2
            elsif (kam = "00001000") then vyst <= "010001";    --3
            elsif (kam = "00010000") then vyst <= "000000";    --4
            elsif (kam = "00100000") then vyst <= "100001";    --5
            elsif (kam = "01000000") then vyst <= "100010";    --6
            elsif (kam = "10000000") then vyst <= "100011";    --7
            elsif (kam = "00000000") then vyst <= "000000";    --00

         else vyst <= "111111";
      end if;



      when s5 =>

            if (kam = "00000001") then vyst <= "010101";       --0
            elsif (kam = "00000010") then vyst <= "010100";    --1
            elsif (kam = "00000100") then vyst <= "010011";    --2
            elsif (kam = "00001000") then vyst <= "010010";    --3
            elsif (kam = "00010000") then vyst <= "010001";    --4
            elsif (kam = "00100000") then vyst <= "000000";    --5
            elsif (kam = "01000000") then vyst <= "100001";    --6
            elsif (kam = "10000000") then vyst <= "100010";    --7
            elsif (kam = "00000000") then vyst <= "000000";    --00

         else vyst <= "111111";
      end if;


      when s6 =>

            if (kam = "00000001") then vyst <= "010110";       --0
            elsif (kam = "00000010") then vyst <= "010101";    --1
            elsif (kam = "00000100") then vyst <= "010100";    --2
            elsif (kam = "00001000") then vyst <= "010011";    --3
            elsif (kam = "00010000") then vyst <= "010010";    --4
            elsif (kam = "00100000") then vyst <= "010001";    --5
            elsif (kam = "01000000") then vyst <= "000000";    --6
            elsif (kam = "10000000") then vyst <= "100001";    --7
            elsif (kam = "00000000") then vyst <= "000000";    --00

         else vyst <= "111111";
      end if;



      when s7 =>

         if (kam = "00000001") then vyst <= "010111";          --0
            elsif (kam = "00000010") then vyst <= "010110";    --1
            elsif (kam = "00000100") then vyst <= "010101";    --2
            elsif (kam = "00001000") then vyst <= "010100";    --3
            elsif (kam = "00010000") then vyst <= "010011";    --4
            elsif (kam = "00100000") then vyst <= "010010";    --5
            elsif (kam = "01000000") then vyst <= "010001";    --6
            elsif (kam = "10000000") then vyst <= "000000";    --7
            elsif (kam = "00000000") then vyst <= "000000";    --00

   else vyst <= "111111";
end if;


when others => NULL;
end case;
patro <= state;
end process OUTPUT_DECODE;
```

Simulace:

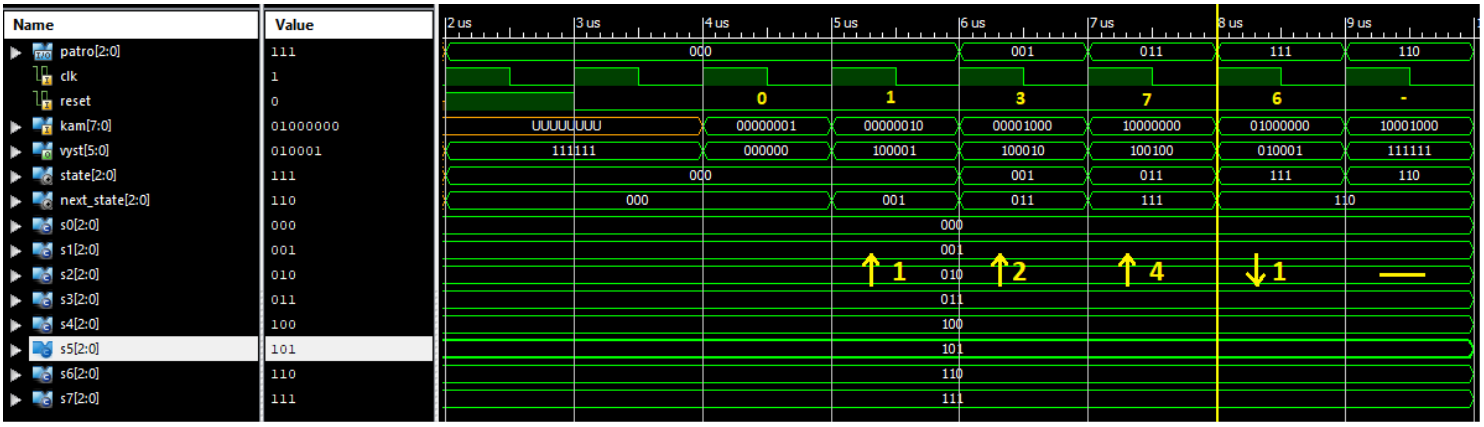
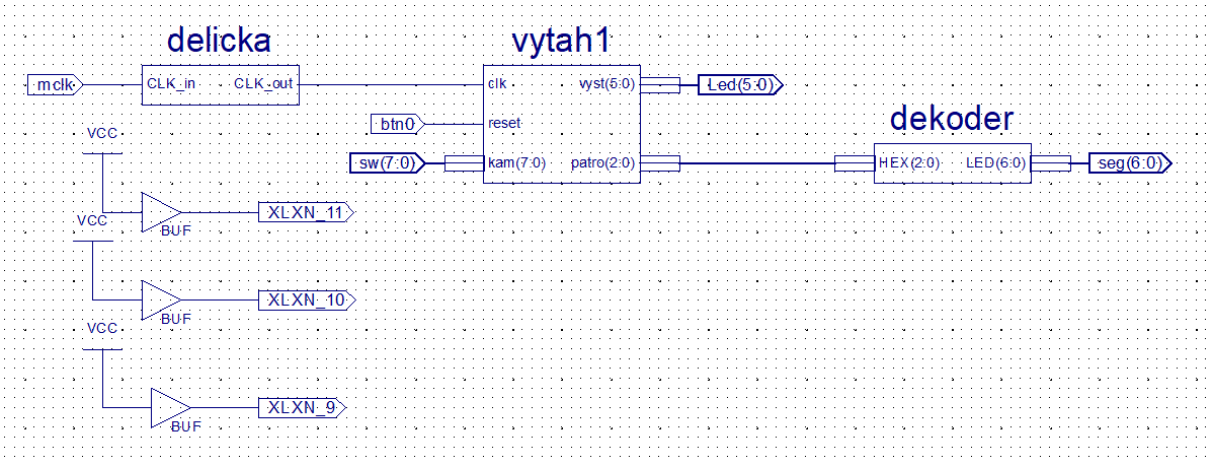


Schéma:



Piny:

```
# clock pins for Basys2 Board
NET "mclk" LOC = "B8"; # Bank = 0, Signal name = MCLK

# Pin assignment for DispCtl
# Connected to Basys2 onBoard 7seg display
NET "seg<0>" LOC = "L14"; # Bank = 1, Signal name = CA
NET "seg<1>" LOC = "H12"; # Bank = 1, Signal name = CB
NET "seg<2>" LOC = "N14"; # Bank = 1, Signal name = CC
NET "seg<3>" LOC = "N11"; # Bank = 2, Signal name = CD
NET "seg<4>" LOC = "P12"; # Bank = 2, Signal name = CE
NET "seg<5>" LOC = "L13"; # Bank = 1, Signal name = CF
NET "seg<6>" LOC = "M12"; # Bank = 1, Signal name = CG
#NET "dp" LOC = "N13"; # Bank = 1, Signal name = DP

NET "an3" LOC = "K14"; # Bank = 1, Signal name = AN3
NET "an2" LOC = "M13"; # Bank = 1, Signal name = AN2
NET "an1" LOC = "J12"; # Bank = 1, Signal name = AN1
#NET "an0" LOC = "F12"; # Bank = 1, Signal name = AN0

# Pin assignment for LEDs
#NET "Led<7>" LOC = "G1" ; # Bank = 3, Signal name = LD7
#NET "Led<6>" LOC = "P4" ; # Bank = 2, Signal name = LD6
NET "Led<5>" LOC = "N4" ;  # Bank = 2, Signal name = LD5
NET "Led<4>" LOC = "N5" ;  # Bank = 2, Signal name = LD4
NET "Led<3>" LOC = "P6" ; # Bank = 2, Signal name = LD3
NET "Led<2>" LOC = "P7" ; # Bank = 3, Signal name = LD2
NET "Led<1>" LOC = "M11" ; # Bank = 2, Signal name = LD1
NET "Led<0>" LOC = "M5" ;  # Bank = 2, Signal name = LD0

# Pin assignment for SWs
NET "sw7" LOC = "N3";  # Bank = 2, Signal name = SW7
NET "sw6" LOC = "E2";  # Bank = 3, Signal name = SW6
NET "sw5" LOC = "F3";  # Bank = 3, Signal name = SW5
NET "sw4" LOC = "G3";  # Bank = 3, Signal name = SW4
NET "sw3" LOC = "B4";  # Bank = 3, Signal name = SW3
NET "sw2" LOC = "K3";  # Bank = 3, Signal name = SW2
NET "sw1" LOC = "L3";  # Bank = 3, Signal name = SW1
NET "sw0" LOC = "P11";  # Bank = 2, Signal name = SW0

#NET "btn3" LOC = "A7";  # Bank = 1, Signal name = BTN3
#NET "btn2" LOC = "M4";  # Bank = 0, Signal name = BTN2
#NET "btn1" LOC = "C11"; # Bank = 2, Signal name = BTN1
NET "btn0" LOC = "G12"; # Bank = 0, Signal name = BTN0

## Pin assignment for PS2
#NET "ps2c"    LOC = "B1"   | DRIVE = 2  | PULLUP ; # Bank = 3, Signal name = PS2C
#NET "ps2d"    LOC = "C3"   | DRIVE = 2  | PULLUP ; # Bank = 3, Signal name = PS2D
```

Zhodnocení:

Úkolem bylo vytvořit funkční výtah o sedmi patrech, který bude rozsvicovat Led diody podle toho, jakým směrem výtah pojede a o kolik pater se posune a výběr patra děláme přes přepínače. Tento úkol mi přišel poněkud jednodušší jelikož jsme podobný úkol již dělali. Program jsem dělal jenom v online nestihl dopsat a musel jsem ho dodělat doma.

Daniel Slowik 4.C