Protokol Výtah

Zadání:

Realizujte Mealyho automat, který vyhodnocuje posun výtahu na základě vstupních informací o zvoleném patře. Sedmisegment ukazuje ve kterém patře se výtah nachází a 5 výstupních Led určuje: Led0 a Led1 směr posuvu, Led3 o 1 patro, Led4 o 2 patra a Led5 o 3 patra. Patra jsou 0-3. Pokud výtah stojí, svítí všechny 3 Led (Led 3, Led4, Led5).

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:

	btn_0	btn_1	btn_2	btn_3
X4	0	0	0	0
X0	1	0	0	0
X1	0	1	0	0
X2	0	0	1	0
X3	0	0	0	1

Vnitřní proměnné:

	q0	q1
S0	0	0
S1	0	1
S2	1	0
S3	1	1

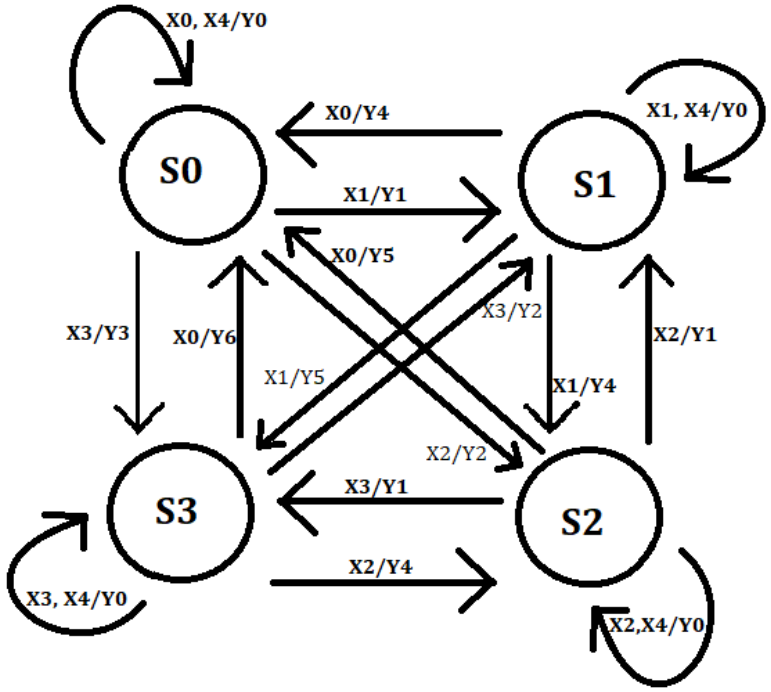
Výstupní proměnné:

	L5	L4	L3	L2	L1	L0
Y0	1	1	1	0	0	0
Y1	0	0	1	0	1	0
Y2	0	1	0	0	1	0
Y3	1	0	0	0	1	0
Y4	0	0	1	0	0	1
Y5	0	1	0	0	0	1
Y6	1	0	0	0	0	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
S0	S0	Y0	S1	Y1	S2	Y2	S3	Y3	S0	Y0
S1	S0	Y4	S1	Y0	S2	Y1	S3	Y2	S1	Y0
S2	S0	Y5	S1	Y4	S2	Y0	S3	Y1	S2	Y0
S3	S0	Y6	S1	Y5	S2	Y4	S3	Y0	S3	Y0

VHDL moduly – programy:

Dělička:

```
library IEEE;
use IEEE.STD_LOGIC_1164.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 (1 downto 0);
            LED:   out   STD_LOGIC_VECTOR (6 downto 0)
    );
end dekoder;

architecture Behavioral of dekoder is

begin

    with HEX SELect
   LED<= "1111001" when "01",    --1
         "0100100" when "10",    --2
         "0110000" when "11",    --3
         "1000000" when others;  --0

end Behavioral;
```

Výtah:

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

entity vytah is
    Port ( patro : inout  STD_LOGIC_VECTOR (1 downto 0);
           clk : in  STD_LOGIC;
           rst : in  STD_LOGIC;
           kam : in  STD_LOGIC_VECTOR (3 downto 0);
           vyst : out  STD_LOGIC_VECTOR (5 downto 0));
end vytah;

architecture Behavioral of vytah is
  signal state, next_state : STD_LOGIC_VECTOR (1 downto 0);

   constant s0 : STD_LOGIC_VECTOR (1 downto 0) := "00";
   constant s1 : STD_LOGIC_VECTOR (1 downto 0) := "01";
   constant s2 : STD_LOGIC_VECTOR (1 downto 0) := "10";
   constant s3 : STD_LOGIC_VECTOR (1 downto 0) := "11";
```

```
20  begin
21     SYNC_PROC: process (clk)
22     begin
23
24     if rising_edge (clk)
25        then if (rst='0')
26           then state <= next_state;
27           else state <= s0;
28        end if;
29     end if;
30     end process SYNC_PROC;
31
32     OUTPUT_DECODE: process (state, kam)
33  Begin
34
35  case (state) is
36     when s0 =>
37        if     (kam = "0001") then vyst <= "111000";
38        elsif (kam = "0010") then vyst <= "001010";
39        elsif (kam = "0100") then vyst <= "010010";
40        elsif (kam = "1000") then vyst <= "100010";
41        else vyst <= "000000";
42        end if;
43     when s1 =>
44        if     (kam = "0001") then vyst <= "001001";
45        elsif (kam = "0010") then vyst <= "111000";
46        elsif (kam = "0100") then vyst <= "001010";
47        elsif (kam = "1000") then vyst <= "010010";
48        else vyst <= "000000";
49        end if;
50     when s2 =>
51           if (kam = "0001") then vyst <= "010001";
52        elsif (kam = "0010") then vyst <= "001001";
53        elsif (kam = "0100") then vyst <= "111000";
54        elsif (kam = "1000") then vyst <= "001010";
55        else vyst <= "000000";
56        end if;
57     when s3 =>
58           if (kam = "0001") then vyst <= "100001";
59        elsif (kam = "0010") then vyst <= "010001";
60        elsif (kam = "0100") then vyst <= "001001";
61        elsif (kam = "1000") then vyst <= "111000";
62        else vyst <= "000000";
63        end if;
64
65     when others => NULL;
66  end case;
67
68  end process OUTPUT_DECODE;
```

```
70  NEXT_STATE_DECODE: process (state, kam)
71  begin
72
73  case (state) is
74     when s0 =>
75        if     (kam = "0001") then next_state <= s0; patro <="00";
76        elsif (kam = "0010") then next_state <= s1; patro <="01";
77        elsif (kam = "0100") then next_state <= s2; patro <="10";
78        elsif (kam = "1000") then next_state <= s3; patro <="11";
79                 else next_state <= s0;
80                 end if;
81
82     when s1 =>
83           if (kam = "0001") then next_state <= s0; patro <="00";
84        elsif (kam = "0010") then next_state <= s1; patro <="01";
85        elsif (kam = "0100") then next_state <= s2; patro <="10";
86        elsif (kam = "1000") then next_state <= s3; patro <="11";
87                 else next_state <= s1;
88                 end if;
89
90     when s2 =>
91           if (kam = "0001") then next_state <= s0; patro <="00";
92        elsif (kam = "0010") then next_state <= s1; patro <="01";
93        elsif (kam = "0100") then next_state <= s2; patro <="10";
94        elsif (kam = "1000") then next_state <= s3; patro <="11";
95                 else next_state <= s2;
96                 end if;
97
98     when s3 =>
99           if (kam = "0001") then next_state <= s0; patro <="00";
100       elsif (kam = "0010") then next_state <= s1; patro <="01";
101       elsif (kam = "0100") then next_state <= s2; patro <="10";
102       elsif (kam = "1000") then next_state <= s3; patro <="11";
103                else next_state <= s3;
104                end if;
105 when others => NULL;
106 end case;
107 patro <= state;
108
109    end process NEXT_STATE_DECODE;
110
111
112  end Behavioral;
```

Simulace:

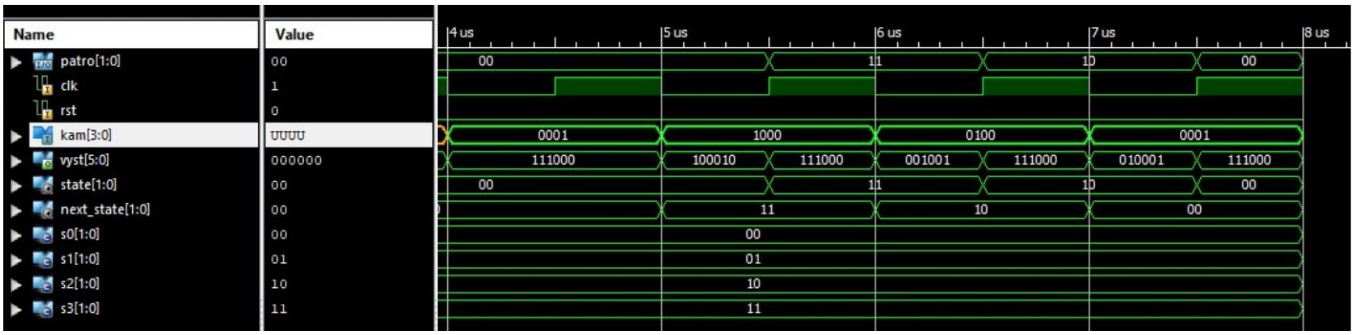
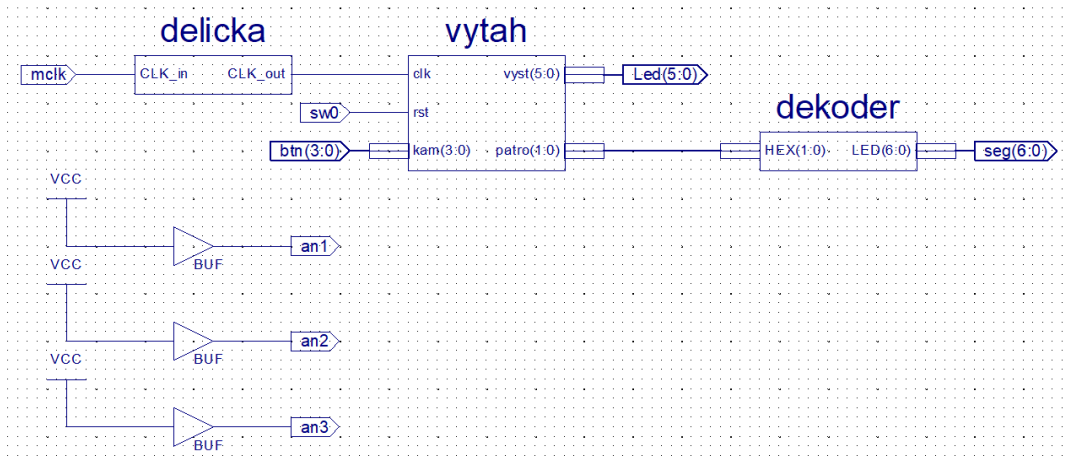


Schéma:



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 "Led0" 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, který bude rozsvicovat Led diody podle toho, jakým směrem výtah pojede a o kolik pater se posune. Tento úkol mi přišel poněkud obtížný a těžce se mi s tímto programem začínalo. Program jsem ve škole nestihl dopsat a musel jsem ho dodělat doma.