Výtah-protokol

Zadáni:

Navrhněte Mealyho automat pro realizaci ovládání a signalizace výtahu. Budova mÁ 0-3 patro. Výtah se pohybuje pomocí tlačíte pater 0-3 a motoru. Na výstupu signalizuje pohyb nahoru nebo dolů(L4, L5) a o kolik pater se posune(L0 o 1 patro, L1 o 2 patra, L3 o 3 patra). Aktuální patro je zobrazeno na sedmisegmentu. Pokud výtah stojí, svítí všechny 3 ledky(L0, L1, L2).

Teoretický rozbor :

Jazyk VHDL - Jazyk VHDL je speciální programovací jazyk pro popis hardwaru. Tento jazyk byl standardizován v roce 1987 a umožňuje popis a simulaci rozsáhlých číslicových systémů. Základní vlastnosti jazyka VHDL:

- Má bohaté vyjadřovací schopnosti.
- Je zde značná nezávislost číslicového systému popsaného tímto jazykem na cílovém prvku, v kterém bude tento systém použit.
- Je možné provádět simulaci chování číslicového systému.

Základním stavebním prvkem jazyka VHDL je tzv. entita, která popisuje vstupní a výstupní signály číslicového systému. Vlastní chování a funkci entity definuje tzv. architektura, která popisuje, jakým způsobem se zpracovávají vstupní signály a generují výstupní signály entity. Architektura je součástí entity a každá entita musí mít alespoň jednu architekturu. Více informací o jazyku VHDL lze získat v literatuře a na internetu.

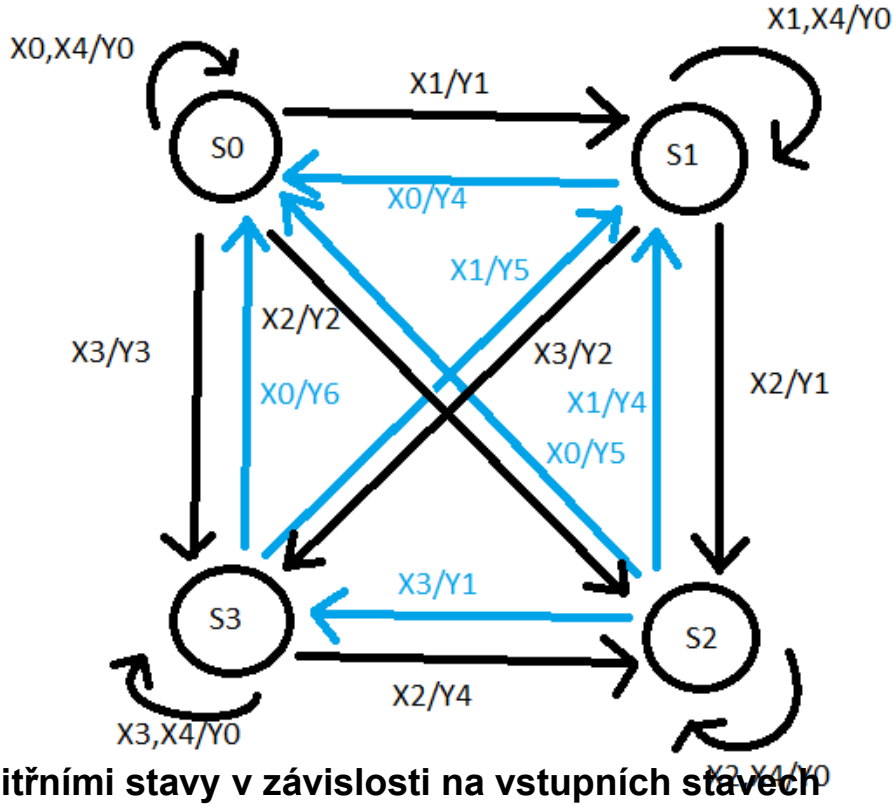
Definice stavů a jejich kódování:

	vstupní		výstupní		vnitřní
	KAM(3:0)		Led(4:0)		Q1 Q0
X0	0 0 0 1	Y0	11100	S0	0 0
X1	0 0 1 0	Y1	00110	S1	0 1
X2	0 1 0 0	Y2	01010	S2	1 0
X3	1 0 0 0	Y3	10010	S3	1 1
X4	0 0 0 0	Y4	00101		
		Y5	01001		
		Y6	10001		

Popis Mealyho automatu:

Výstup je generován na základě příchozího vstupu i momentálního stavu, ve kterém se automat nachází. To znamená, že 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. Tímto Mealyho automat připomíná synchronní komunikaci: Nejen že reaguje na hranu vstupního signálu, ale jakmile ho zpracuje a dosáhne dalšího stavu, jednou vygeneruje výstupní hodnotu, puls výstupního signálu, a pak už žádný výstup neposkytuje; zase až do další vstupní hodnoty předložené ke zpracování. Totiž nejen, že jsou *stavy* Mealyho stroje podmnožinou kartézského součinu množiny (předešlých) stavů a *vstupní* abecedy, ale i jeho *výstupy* jsou podmnožinou kartézského součinu stavů a *výstupní* abecedy.

Orientovaný graf:



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

	X0	X1	X2	X3	X4
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

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;
```

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;
```

Hlavní program vytah_main:

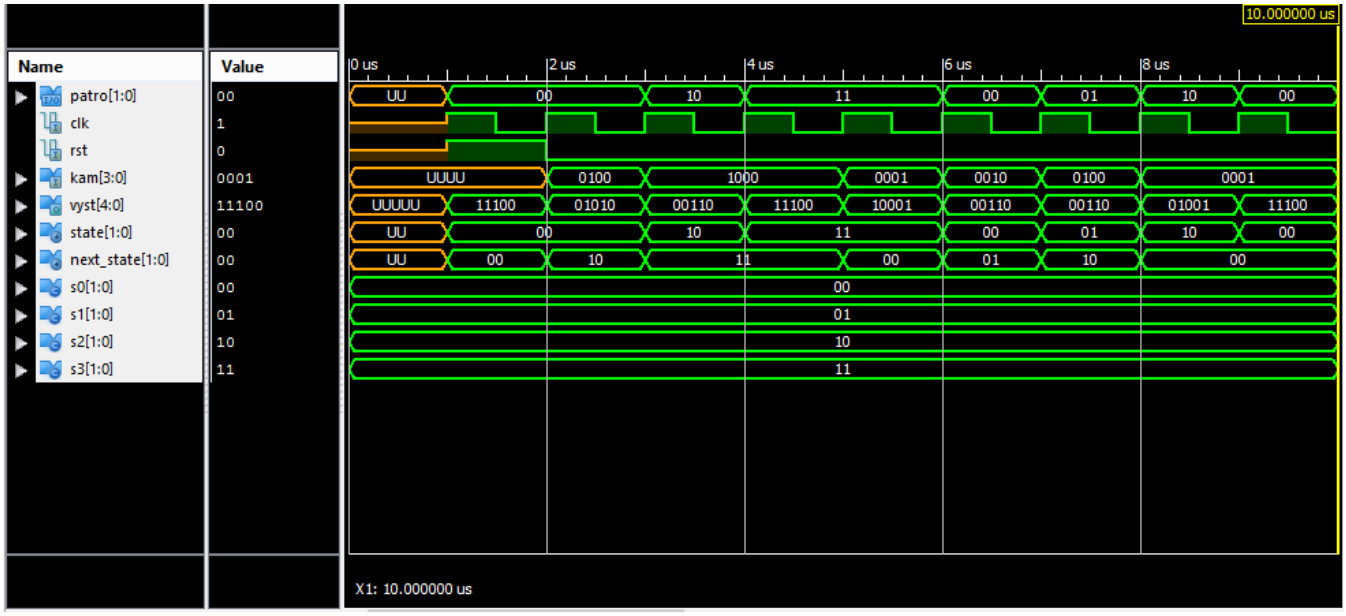
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity vytah_main 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 (4 downto 0));
```

```
end vytah_main;

architecture Behavioral of vytah_main 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";

begin
 SYNC_PROC: process (clk, rst)
  begin
                    if(rst='1') then
       state <= s0;

                    elsif rising_edge(clk) then
       state <= next_state;
    end if;
  end process;

  OUTPUT_DECODE: process (state, kam)
  begin
                    case (state) is
                      when s0 =>
                                                  if (kam = "0001") then vyst <= "11100";
                                                  elsif (kam = "0010") then vyst <= "00110";
                                                  elsif (kam = "0100") then vyst <= "01010";
                                                  elsif (kam = "1000") then vyst <= "10010";
                                                  else      vyst <= "11100";
                                                  end if;

                          when s1 =>
                                                  if (kam = "0001") then vyst <= "00101";
                                                  elsif (kam = "0010") then vyst <= "11100";
                                                  elsif (kam = "0100") then vyst <= "00110";
                                                  elsif (kam = "1000") then vyst <= "01010";
                                                  else      vyst <= "11100";
```

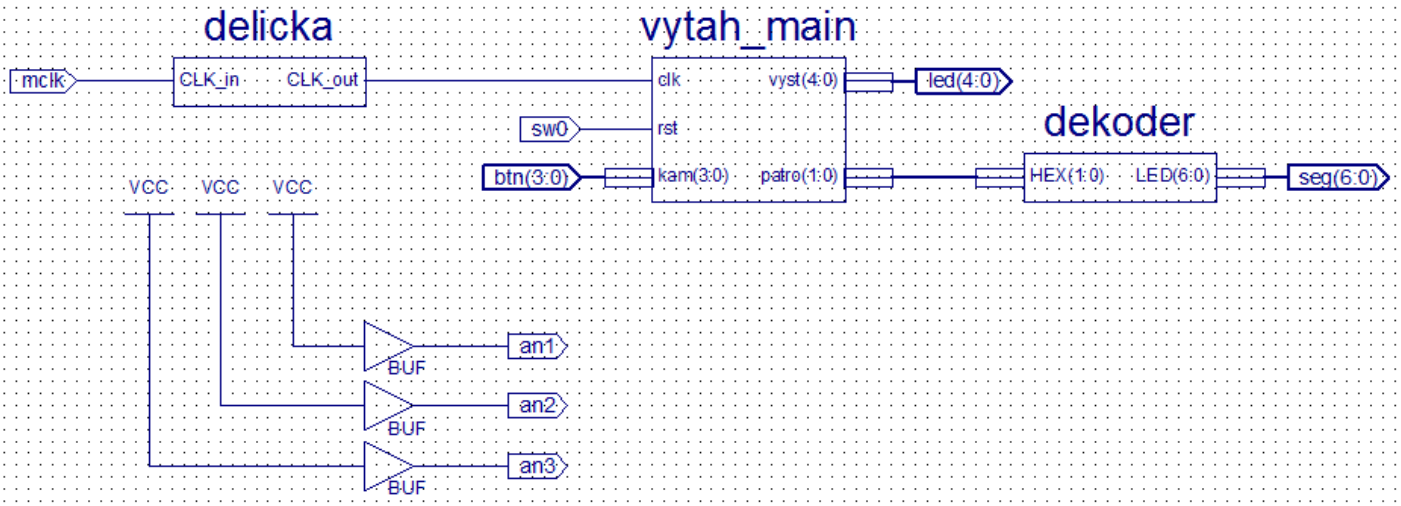
```
                                        end if;

                        when s2 =>
                                        if (kam = "0001") then vyst <= "01001";
                                        elsif (kam = "0010") then vyst <= "00101";
                                        elsif (kam = "0100") then vyst <= "11100";
                                        elsif (kam = "1000") then vyst <= "00110";
                                        else      vyst <= "11100";
                                        end if;

                        when s3 =>
                                        if (kam = "0001") then vyst <= "10001";
                                        elsif (kam = "0010") then vyst <= "01001";
                                        elsif (kam = "0100") then vyst <= "00110";
                                        elsif (kam = "1000") then vyst <= "11100";
                                        else      vyst <= "11100";
                                        end if;


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

NEXT_STATE_DECODE: process (state, kam)
begin
case (state) is
                        when s0 =>
                        if (kam = "0001") then next_state <= s0; patro <="00";
                        elsif (kam = "0010") then next_state <= s1; patro <="01";
                        elsif (kam = "0100") then next_state <= s2; patro <="10";
                        elsif (kam = "1000") then next_state <= s3; patro <="11";
                        else next_state <= s0;
                        end if;

                        when s1 =>
                        if (kam = "0001") then next_state <= s0; patro <="00";
                        elsif (kam = "0010") then next_state <= s1; patro <="01";
```

```
elsif (kam = "0100") then next_state <= s2; patro <="10";
elsif (kam = "1000") then next_state <= s3; patro <="11";
else next_state <= s1;
end if;

when s2 =>
if (kam = "0001") then next_state <= s0; patro <="00";
elsif (kam = "0010") then next_state <= s1; patro <="01";
elsif (kam = "0100") then next_state <= s2; patro <="10";
elsif (kam = "1000") then next_state <= s3; patro <="11";
else next_state <= s2;
end if;

when s3 =>
if (kam = "0001") then next_state <= s0; patro <="00";
elsif (kam = "0010") then next_state <= s1; patro <="01";
elsif (kam = "0100") then next_state <= s2; patro <="10";
elsif (kam = "1000") then next_state <= s3; patro <="11";
else next_state <= s3;
end if;

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

Simulace:



Celkové schéma:



Výpis pinů:

```
# 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:

Za úkol jsme dostali vytvořit v programu Xilinx výtah, který jezdí od přízemí až do 3 patra.
Jako první jsem pomocí nápovědy vytvořil děličku, následně dekodér, a podle nápovědy Mealyho automatu jsem vytvořil i hlavní program výtah_main. U každého modulu jsem zkontroloval syntaktické chyby, a po opravě jsem vytvořil schématické značky. Nasimuloval jsem výtah i v simulačním prostředí, a fungoval bez problému. Tento program jsem stihl i ve škole otestovat na přípravku, a vše bylo v pořádku. Program sám o sobě je jeden z náročnějších, a vyžaduje znalosti programovacího jazyka a zkušenosti.

Šimon Nieslanik 4.C