1.

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

2.

Spartan

Spartan je lehká vývojová deska FGA, je založenqa na řipu Xilinx Spartan-7
Můžete použít s Arduinem k ovládání LCD a fotoaparátu nebo jako samostatnou vývojovou desku FPGA.

Místo makrobuňek obsahují logické bloky
Log. Bloky jsou navzájem propojeny globální propojovací maticí

Obsahuje SRAM

VHDL

Používá se pro návrh a simulaci digitálních integrovaných obvodů,
například programovatelných hradlových polí (CPLD,FPGA) nebo různých zákaznických obvodů(ASIC)

Jazyk VHDL může být požit i jako paralelní programovací jazyk.

3.

Vstupní stavy				
	btn3	btn2	btn1	btn0
x4	0	0	0	0
x0	0	0	0	1
x1	0	0	1	0
x2	0	1	0	0
x3	1	0	0	0

Vnitřní stavy			
	Q1	Q0	
s0	0	0	...0 Patro
s1	0	1	...1 Patro
s2	1	0	...2 Patro
s3	1	1	...3 Patro

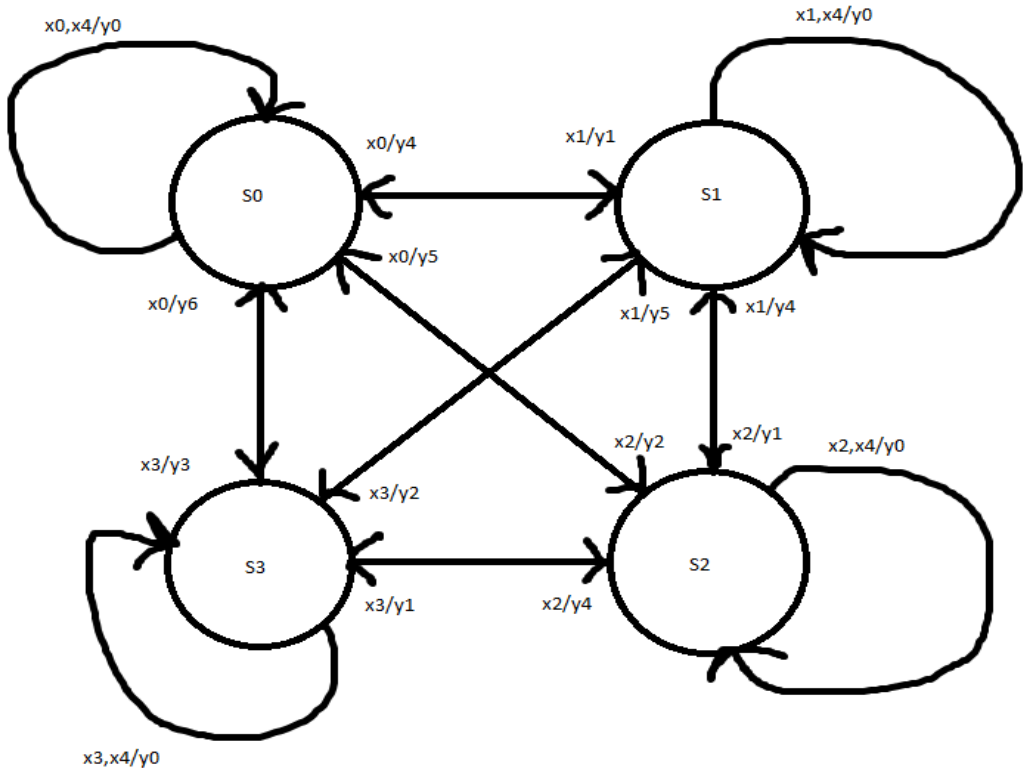
Výstupní stavy							
	L5	L4	L3	L2	L1	L0	
y0	0	0	0	1	1	1	
y1	0	1	0	0	0	1	...Nahoru o 1 patro
y2	0	1	0	0	1	0	...Nahoru o 2 patra
y3	0	1	0	1	0	0	...Nahoru o 3 patra
y4	1	0	0	0	0	1	...Dolu o 1 patro
y5	1	0	0	0	1	0	...Dolu o 2 patra
y6	1	0	0	1	0	0	...Dolu o 3 patra

4.

y= f(S) Moore... Závislý na vnitřním stavu

y= f(X,S) Mealy... Závislý na vnitřním stavu a vstupu

5.



6.

	x0	x1	x2	x3	y
S0	s0	s1	s2	s3	y0 - y3
s1	s0	s1	s2	s3	y0 - y2, y4
s2	s0	s1	s2	s3	y0 - y1,y4 -y5
s3	s0	s1	s2	s3	y0, y4 - y6

7.

DEKODER

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

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity dekoder is
    Port ( stavy : in  STD_LOGIC_VECTOR (1 downto 0);
           LED : out  STD_LOGIC_VECTOR (6 downto 0));
end dekoder;

architecture Behavioral of dekoder is

begin

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

end Behavioral;
```

DELICKA

```
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;
```

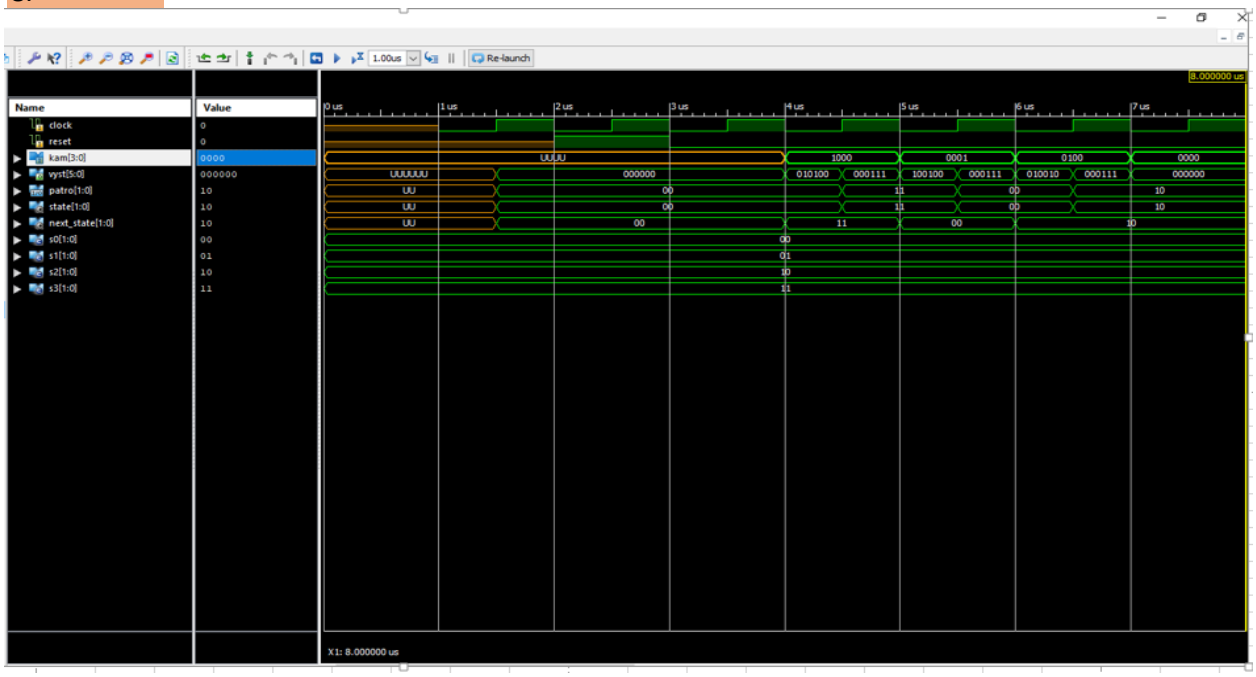
Hl.program

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


entity main is
   Port ( clock : in  STD_LOGIC;
          reset : in  STD_LOGIC;
          kam : in  STD_LOGIC_VECTOR (3 downto 0);
          vyst : out  STD_LOGIC_VECTOR (5 downto 0);
          patro : inout  STD_LOGIC_VECTOR (1 downto 0));
end main;

architecture Behavioral of 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 (clock)
  begin

  if rising_edge (clock)
                        then if (reset='0')
                                then state <= next_state;
                                else state <= s0;
                        end if;
                end if;
  end process SYNC_PROC;


  OUTPUT_DECODE: process (state, kam)
Begin

case (state) is
                when s0 =>
                        if (kam = "(  "000111";
                        elsif (kam = "0010") then vyst <= "010001";
                        elsif (kam = "0100") then vyst <= "010010";
                        elsif (kam = "1000") then vyst <= "010100";
                        else vyst <= "000000";
                        end if;
                when s1 =>
                        if (kam = "(  "100001";
                        elsif (kam = "0010") then vyst <= "000111";
```

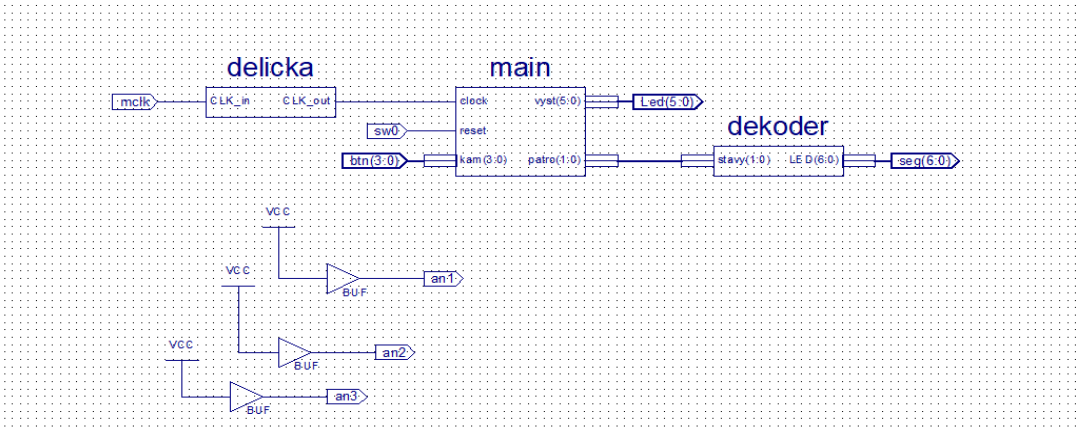
```
                elsif (kam = "0100") then vyst <= "010001";
                elsif (kam = "1000") then vyst <= "010010";
                else vyst <= "000000";
                end if;
        when s2 =>
          if (kam =   "100010";
                elsif (kam = "0010") then vyst <= "100001";
                elsif (kam = "0100") then vyst <= "000111";
                elsif (kam = "1000") then vyst <= "010001";
                else vyst <= "000000";
                end if;
        when s3 =>
          if (kam =   "100100";
                elsif (kam = "0010") then vyst <= "100010";
                elsif (kam = "0100") then vyst <= "100001";
                elsif (kam = "1000") then vyst <= "000111";
                else vyst <= "000000";
                end if;

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

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;
```

8.



9.



10.

```
# 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
```

11.

Úkol byl naprogramovat mealyho automat.
Jedná se o výtah s 3 patry
Je možné přepínat mezi těmito patry a aktualní patro se ukáže na sedmisegmentu
Pokud zmáčkneme číslo patra ve kterém se výtah aktuálně náchází rozsvíti se příslušné 3 ledky.

Při práci s ISE project nastály 2 problémy z simulací.
1 problém byl v typu simulace změně typu z ModelSim na ISIM simulace fungovala.
Dále mi nastala chyba při spuštění simulace mi antivirus zaznamená chybu a vypne mi simulaci, tento problém jsem vyřešil vypnutím antiviru.

Mimo tyto zmíněné problémy nenastaly žádné další nepříjemnosti.