1.

Navrhněte Mealyho automat, který řeší otevírání skříňky na vlakovém nádraží. Cena je 5Kč, vhazované mince jsou 1,2 a 5-koruna. Automat vrací 1Kč pouze v případě, že zákazník vhodil 4Kč a následně 2-korunovou minci.

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			
	1 Kč	2 Kč	5 Kč
x0	0	0	0
x1	1	0	0
x2	0	1	0
x3	0	0	1

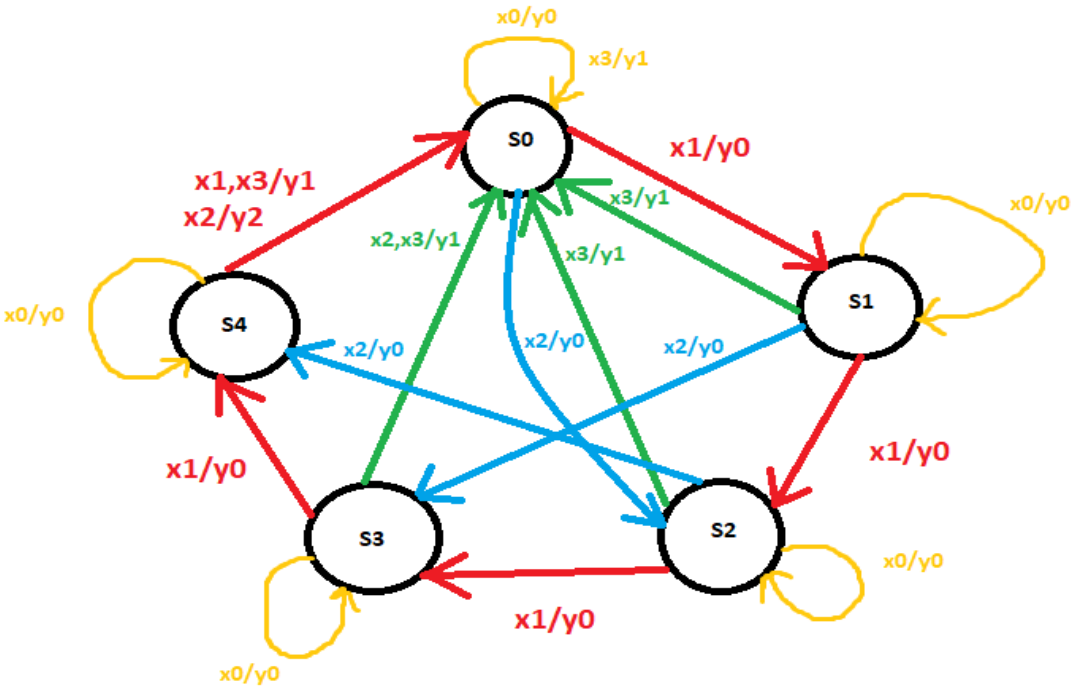
vnitřní stavy				
	Q1	Q2	Q3	
S0	0	0	0	...0 korun
S1	0	0	1	...1 koruna
S2	0	1	0	...2 koruny
S3	0	1	1	...3 koruny
s4	1	0	0	...4 koruny

výstupní stavy		
	otevřít	vrátit
y0	0	0
y1	1	0
y2	1	1

4.

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

5.



6.

	x0/y	x1/y	x2/y	x3/y
s0	s0/y0	s1/y0	s2/y0	s1/y0
s1	s1/y0	s2/y0	s3/y0	s1/y0
s2	s2/y0	s3/y0	s4/y0	s1/y0
s3	s3/y0	s4/y0	s0/y1	s1/y0
s4	s4/y0	s0/y1	s0/y2	s1/y0

7.

Dekoder

```
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
     "0010010" when "101",   --5
     "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;
```

Hlavní program

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


entity main is
    Port ( clock : in  STD_LOGIC;
           reset : in  STD_LOGIC;
           mince : in  STD_LOGIC_VECTOR (2 downto 0);
           otevrit : out  STD_LOGIC_VECTOR (1 downto 0);
           stav : inout  STD_LOGIC_VECTOR (2 downto 0));
end main;

architecture Behavioral of main is

signal state, next_state : std_logic_vector(2 downto 0);

   constant s0 : std_logic_vector(2 downto 0)  := "000";
   constant s1 : std_logic_vector(2 downto 0)  := "001";
   constant s2 : std_logic_vector(2 downto 0)  := "010";
   constant s3 : std_logic_vector(2 downto 0)  := "011";
   constant s4 : std_logic_vector(2 downto 0)  := "100";

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;
```

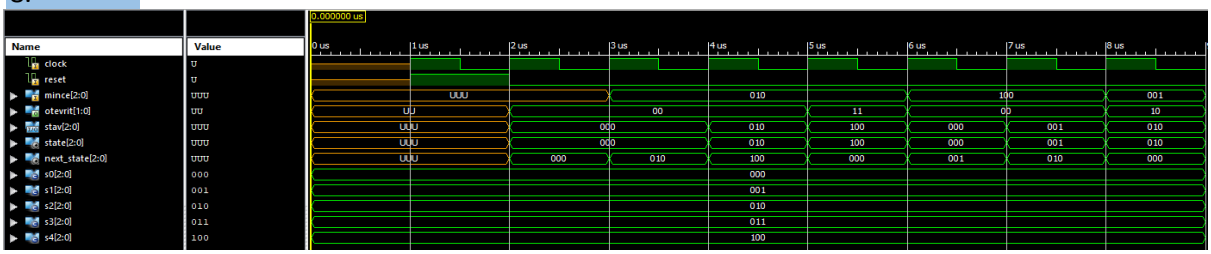
```
   OUTPUT_DECODE: process (mince,state)
Begin

case (state) is
   when s0 =>
      if (mince = "000") then otevrit <= "00";
      elsif (mince = "100") then otevrit <= "00";
      elsif (mince = "010") then otevrit <= "00";
      elsif (mince = "001") then otevrit <= "10";
      else otevrit <= "00";
      end if;
   when s1 =>
      if (mince = "000") then otevrit <= "00";
      elsif (mince = "100") then otevrit <= "00";
      elsif (mince = "010") then otevrit <= "00";
      elsif (mince = "001") then otevrit <= "10";
      else otevrit <= "00";
      end if;
   when s2 =>
         if(mince = "000") then otevrit <= "00";
      elsif (mince = "100") then otevrit <= "00";
      elsif (mince = "010") then otevrit <= "00";
      elsif (mince = "001") then otevrit <= "10";
      else otevrit <= "00";
      end if;
   when s3 =>
         if(mince = "000") then otevrit <= "00";
      elsif (mince = "100") then otevrit <= "00";
      elsif (mince = "010") then otevrit <= "10";
      elsif (mince = "001") then otevrit <= "10";
      else otevrit <= "00";
      end if;
   when s4 =>
         if(mince = "000") then otevrit <= "00";
      elsif (mince = "100") then otevrit <= "10";
      elsif (mince = "010") then otevrit <= "11";
      elsif (mince = "001") then otevrit <= "10";
      else otevrit <= "00";
      end if;
```

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

NEXT_STATE_DECODE: process (state, mince)
begin

case (state) is
   when s0 =>
      if (mince = "000") then next_state <= s0; stav <="000";
      elsif (mince = "100") then next_state <= s1; stav <="001";
      elsif (mince = "010") then next_state <= s2; stav <="010";
      elsif (mince = "001") then next_state <= s0; stav <="000";
               else next_state <= s0;
               end if;
```

```
when s1 =>
            if (mince = "000") then next_state <= s1; stav <="001";
   elsif (mince = "100") then next_state <= s2; stav <="010";
   elsif (mince = "010") then next_state <= s3; stav <="011";
   elsif (mince = "001") then next_state <= s0; stav <="000";
            else next_state <= s1;
            end if;

when s2 =>
            if (mince = "000") then next_state <= s2; stav <="010";
   elsif (mince = "100") then next_state <= s3; stav <="011";
   elsif (mince = "010") then next_state <= s4; stav <="100";
   elsif (mince = "001") then next_state <= s0; stav <="000";
            else next_state <= s2;
            end if;

when s3 =>
            if (mince = "000") then next_state <= s3; stav <="011";
   elsif (mince = "100") then next_state <= s4; stav <="100";
   elsif (mince = "010") then next_state <= s0; stav <="000";
   elsif (mince = "001") then next_state <= s0; stav <="000";
            else next_state <= s3;
            end if;
```

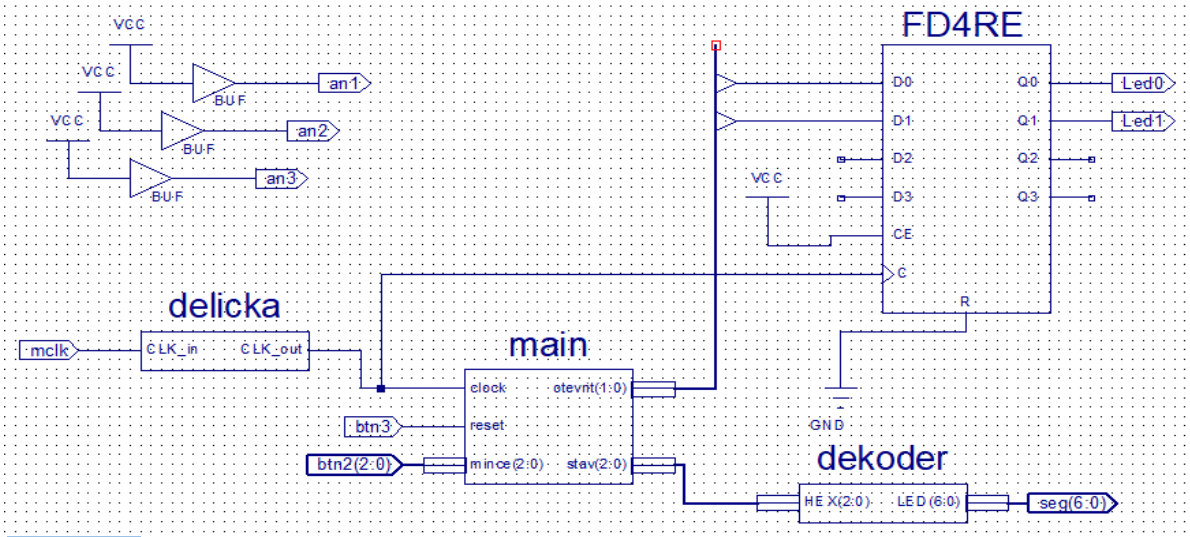
```
 when s4 =>
            if (mince = "000") then next_state <= s4; stav <="100";
    elsif (mince = "100") then next_state <= s0; stav <="000";
    elsif (mince = "010") then next_state <= s0; stav <="000";
    elsif (mince = "001") then next_state <= s0; stav <="000";
            else next_state <= s4;
            end if;

when others => NULL;
end case;
stav <= state;

end process NEXT_STATE_DECODE;


end Behavioral;
```

8.



Simulaci se mi nepodařilo udělat čitelnější tak jsem jí poslal s protokolem

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 skříňku
pro zaplacení lístku je nutné zaplatit 5 korun
Je možné vhazovat 1,2 a 5 korun, celkový počet se pak následně ukáže na sedmisegmentu
Dále byla podmínka že pokuď je ve skříňce 4 koruny a vložíme další 2 automat nám vrátí 1 korunu.
v jiných případech automat nevrací nic

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,vypne mi simulaci,
tento problém jsem vyřešil vypnutím antiviru.

Mimo tyto zmíněné problémy nenastaly žádné další nepříjemnosti.