# Protokol Garáž

**Zadaní:**

Moore automat pro garáž pro 5 aut, indikace počtu na sedmisegmentu a semafor svítí červeně u plné garáže.

**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:

| | X0 | X1 | X2 | X3 |
|---|---|---|---|---|
| p | 0 | 1 | 0 | 1 |
| o | 0 | 0 | 1 | 1 |

Vnitřní stavy:

| S0 | S1 | S2 | S3 | S4 | S5 |
|---|---|---|---|---|---|
| 000 | 001 | 010 | 011 | 100 | 101 |

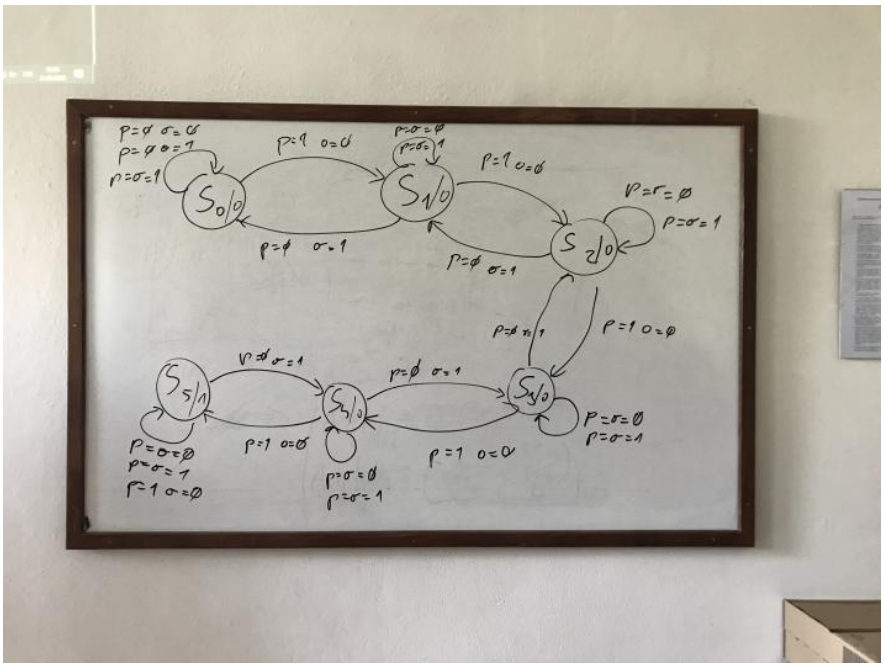
Výstupní:

| Y0 | Y1 |
|---|---|
| 0 | 1 |

**Popis automatu Moore:**

Moorův automat je jednoduché zařízení s konečným počtem vnitřních stavů, mezi kterými se přechází na základě vstupních symbolů. Každý vnitřní stav má definovaný právě jednu hodnotu na výstupu. Automat musí mít dále definovaný výchozí vnitřní stav, ve kterém se nachází před zadáním prvního vstupního symbolu a pravidla pro přechody mezi jednotlivými stavy.

**Orientovaný graf:**




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

| | X0 | X1 | X2 | X3 | Y |
|---|---|---|---|---|---|
| S0 | S0 | S1 | S0 | S0 | Y0 |
| S1 | S1 | S2 | S0 | S1 | Y0 |
| S2 | S2 | S3 | S1 | S2 | Y0 |
| S3 | S3 | S4 | S2 | S3 | Y0 |
| S4 | S4 | S5 | S3 | S4 | Y0 |
| S5 | S5 | S5 | S4 | S5 | Y1 |

**VHDL moduly – programy:**

Dělička:

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3
4  entity dekoder is
5      Port ( HEX : in  STD_LOGIC_VECTOR (2 downto 0);
6             LED : out  STD_LOGIC_VECTOR (6 downto 0));
7  end dekoder;
8
9  architecture Behavioral of dekoder is
10
11 begin
12
13     with HEX SELect
14    LED<= "1111001" when "001",   --1
15          "0100100" when "010",   --2
16          "0110000" when "011",   --3
17          "0011001" when "100",   --4
18          "0010010" when "101",   --5
19          "1000000" when others;   --0
20
21 end Behavioral;
```

Dekodér:

```
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;
```

Garáž:

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



entity garaz_skr is
   Port (  p : in STD_LOGIC;
           o : in STD_LOGIC;
      reset: in STD_LOGIC;
      clk : in STD_LOGIC;
      stav : inout STD_LOGIC_VECTOR(2 downto 0);
      semafor : out STD_LOGIC);
end garaz_skr;

architecture Behavioral of garaz_skr 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";
   constant S5 : std_logic_vector(2 downto 0) := "101";
begin
   SYNCH_PROCES: process(clk, reset)
   begin
      if (reset = '1') then
         state <= S0;
      elsif rising_edge(clk) then
         state <= next_state;
      end if;
   end process SYNCH_PROCES;

   ZAK_VYSTUP: process(state)
   begin
      case (state) is
         when S5 =>
            semafor <= '1';
         when others =>
            semafor <= '0';
      end case;
   end process ZAK_VYSTUP;

   ZAK_STAVU: process (p, o, state)
begin
   case (state) is
      when S0=>
         if (p= '1' AND o= '0') then
             next_state <= S1;
            else
             next_state <=S0;
            end if;

      when S1 =>
          if (p= '1' AND o= '0') then
             next_state <= S2;
            elsif (p= '0' AND o= '1') then
             next_state <= S0;
            else
             next_state <=S1;
            end if;
      when S2 =>
          if (p= '1' AND o= '0') then
             next_state <= S3;
            elsif (p= '0' AND o= '1') then
             next_state <= S1;
            else
             next_state <=S2;
            end if;
      when S3 =>
          if (p= '1' AND o= '0') then
             next_state <= S4;
            elsif (p= '0' AND o= '1') then
             next_state <= S2;
            else
             next_state <=S3;
            end if;
      when S4 =>
          if (p= '1' AND o= '0') then
             next_state <= S5;
            elsif (p= '0' AND o= '1') then
             next_state <= S3;
            else
             next_state <=S4;
            end if;
      when S5 =>
          if (p= '0' AND o= '1') then
             next_state <= S4;
            else
             next_state <=S5;
            end if;
      when others => NULL;

      end case;

      stav <= state;

   end process ZAK_STAVU;

end Behavioral;
```

**Simulace:**

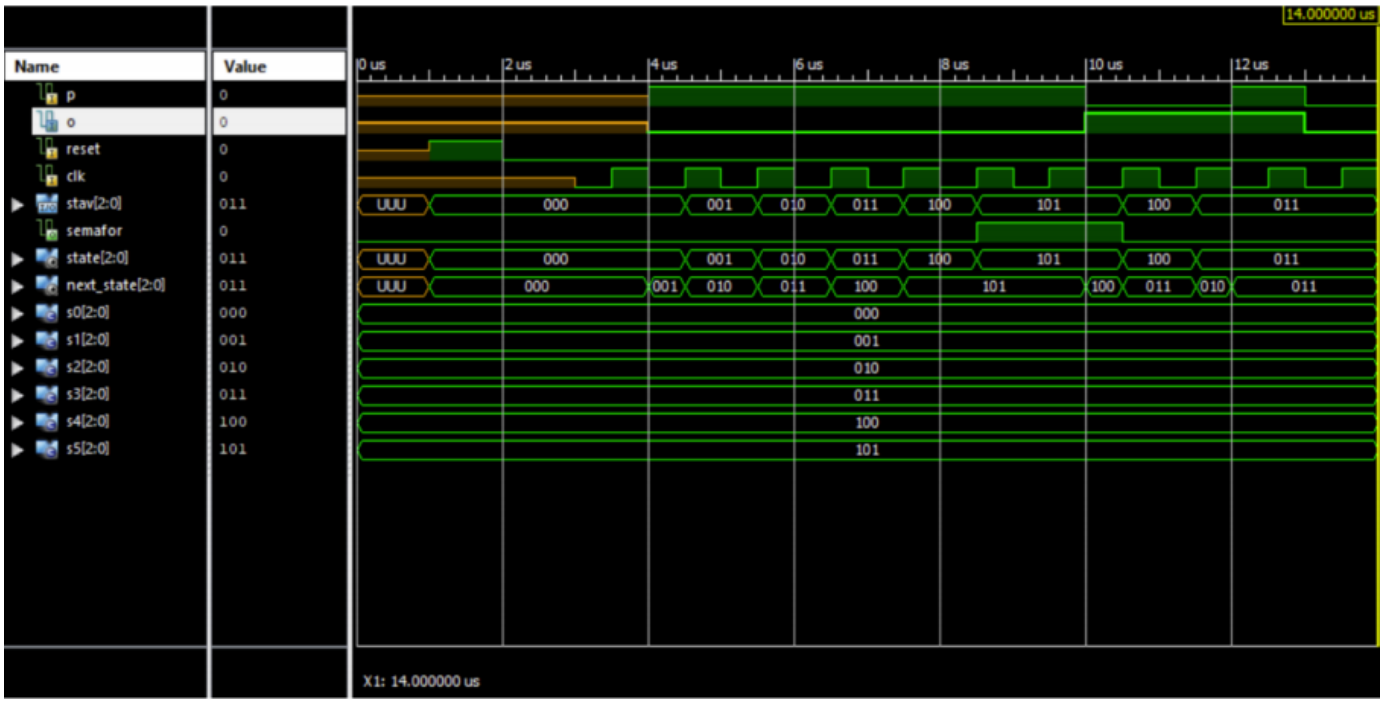
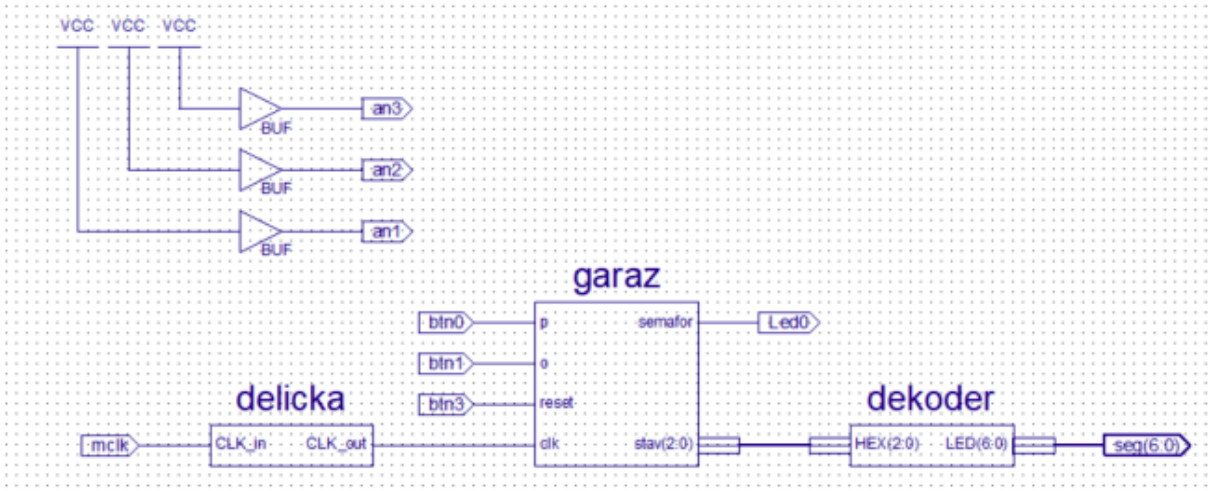



**Schéma:**



**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:**

Projekt jsem zprvu nepochopil, ale poté co jsem dokázal udělat case S0, jsem vše pochopil a dodělal jsem zbytek.