Protokol MIT Garaz

Protokol

1) zadání

2) teor. rozbor Spartan a VHDL

3) definice stavů a jejich kódování

4) popis automatu Moore nebo Mealy

5) orientovaný graf

6) tabulky přechodů mezi vnitřními stavy v závislosti na vstupních stavech, tabulky výstupů

7) VHDL moduly - programy

8) simulace

9) celkové schémaá

10) výpis pinů

11) zhodnocení

1) Zadání: Vytvoření projektu Garáž pro 5 aut, když jsou všechny místa obsazené tak svítí semafor.

2) teor. Rozbor

Jazyk VHDL (Very High Speed Integrated Circuits Hardware Description Language) je spolu s jazykem Verilog HDL jedním z nejpoužívanějším jazykům pro popis hardwarových struktur hradlových polí. Umožňuje návrh jak logických tak i sekvenčních struktur a jeho hlavní výhodou je jeho univerzálnost. Konečná implementace navržené struktury je závislá až na kompilaci VHDL kódu. Tzn., že pomocí tohoto jazyka lze provádět návrhy pro hradlová pole většiny výrobců (Xiling, Altera, Lattice apod.) a následně použít vhodný kompilátor, jehož volná verze je obvykle dostupná na internetových stránkách výrobců. Návrh struktury je možné (a vhodné) rozdělit na samostatné bloky, které jsou následně spojeny ve finálním modulu pomocí objektu typu signál, který většinou odpovídá reálnému elektrickému signálu. Tento postup je v podstatě adekvátní propojování skutečných obvodů v reálu.

FPGA obvody dnes nacházejí uplatnění v široké škále aplikací díky své programovatelnosti, snadnému návrhu, flexibilitě, neustále klesajícím cenám a zvolna se snižující spotřebě energie vlastním čipem. Typické použití je v oblasti menších sérií navrhovaných zařízení, kdy se nevyplatí návrh zákaznického integrovaného obvodu a současně konvenční řešení systému s procesorem už není vhodné. Další aplikace můžeme nalézt například v oblasti prototypování složitějších zákaznických integrovaných obvodů.

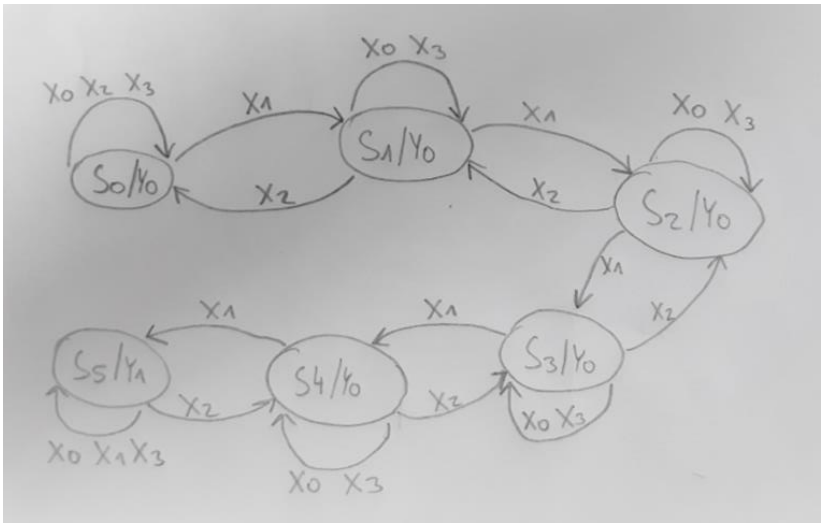
3)

vstupní		vnitřní	výstupní
p	o	stavy	semafor
X0	0 0	S0 000	y0 0
X1	1 0	S1 001	y1 1
X2	0 1	S2 010	
X3	1 1	S3 011	
		S4 100	
		S5 101	

4) popis Moore automatu

Automat typu Moore si lze představit jako 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. Výstupní funkce jsou tedy funkcemi pouze vnitřního stavu.

5) Orientační graf



6) Tabulka

	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

7) VHDL moduly

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 ( LED : out  STD_LOGIC_VECTOR (6 downto 0);
           HEX : in  STD_LOGIC_VECTOR (2 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;
```

Dělička

```
----------------------------------------------------------
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;
```

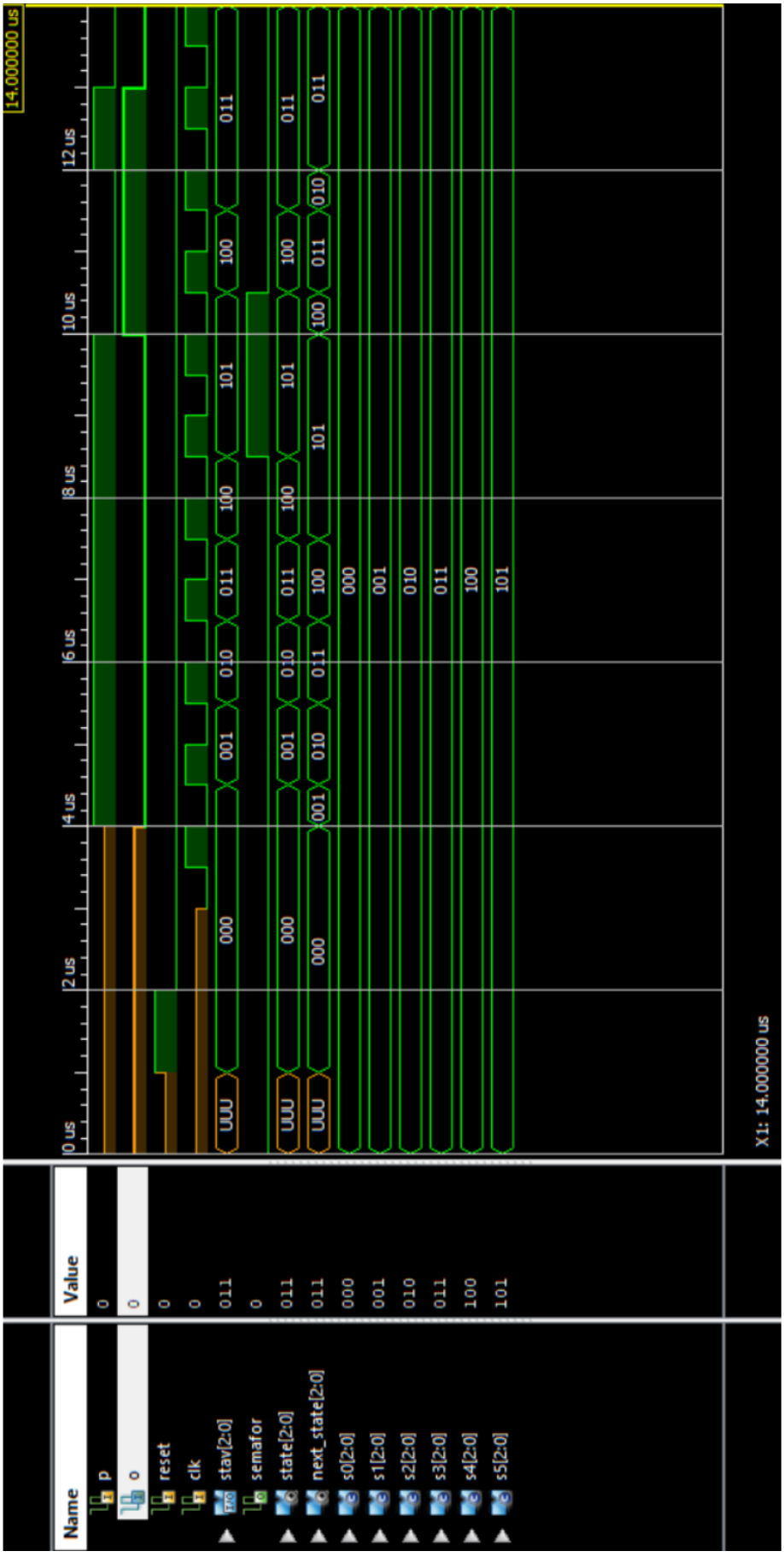
Garaz_main

```
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 garaz 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;

architecture Behavioral of garaz 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

SYNCHRONIZACNI_PROCES: process (clk, reset)
   begin
         if (reset = '1') then
            state <= S0;
         elsif rising_edge(clk) then
            state <= next_state;
         end if;
   end process SYNCHRONIZACNI_PROCES;
```

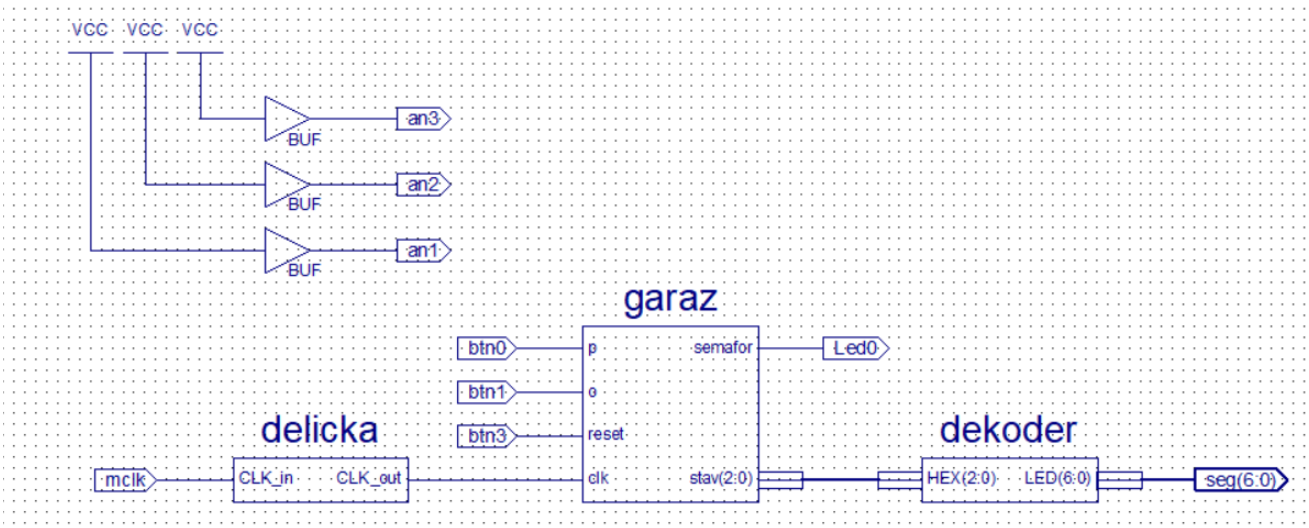
```
ZAKODOVANI_VYSTUPU: process (state)
   begin
       case (state) is
         when S5=>
                                semafor<= '1';

         when others=>
                                semafor<='0';
                        end case;
   end process ZAKODOVANI_VYSTUPU;

ZAKODOVANI_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 ZAKODOVANI_STAVU;


end Behavioral;
```

8) Simulace


Name
Value
p
o
reset
clk
stav[2:0]
semafor
state[2:0]
next_state[2:0]
s0[2:0]
s1[2:0]
s2[2:0]
s3[2:0]
s4[2:0]
s5[2:0]
0 us
2 us
4 us
6 us
8 us
10 us
12 us
14.000000 us
X1: 14.000000 us

9) Schema



10) 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 "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) Zhodnocení

Projekt garáž mi na Spartanu 3E zprvu nefungoval ,ale zjistil jsem ,že chyba byla v neodkomentování mclk. A poté už HW fungoval jak měl.

Návrh na zlepšení je, že když už je poslední volné místo tak by mohla blikat ledka pro signalizaci.

Marián Wojnar 4.C