Tomáš Konvičný 28.10.2020

Protokol – Garáž

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 schéma

10) výpis pinů

11) zhodnocení

1) Zadání:

Navrhněte Moorův automat realizující garáž pro 5 aut. Počet aut v garáži je zobrazen na sedmisegmentovém displeji. Plná garáž je signalizována rozsvícením LED diody. Příjezd a odjezd simulujete tlačítky btn. Vynulování stavu (reset) realizujete přepínačem sw.

2) Teoretický rozbor

Jazyk VHDL (VHSIC Hardware Description Language) je jazyk určený pro popis digitálního hardware. Zkratka VHSIC znamená Very High Speed Integrated Circuits. Jazyk VHDL umožňuje popsat jak strukturu obvodu (tj. zapojení z hradel, bloků apod.), tak i chování obvodu. Jde o jazyk se silnou kontrolou (nejen typovou), kdy je maximální snaha odhalit chyby návrháře již na úrovni překladu zdrojového textu. Vzorem při vývoji byl jazyk ADA, resp. Pascal (na rozdíl od jazyka Verilog, který je odvozen od jazyka C). Jazyk umožňuje definovat typy dat (i generické), proměnné, paralelní procesy.

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) Definice stavů a jejich kódování

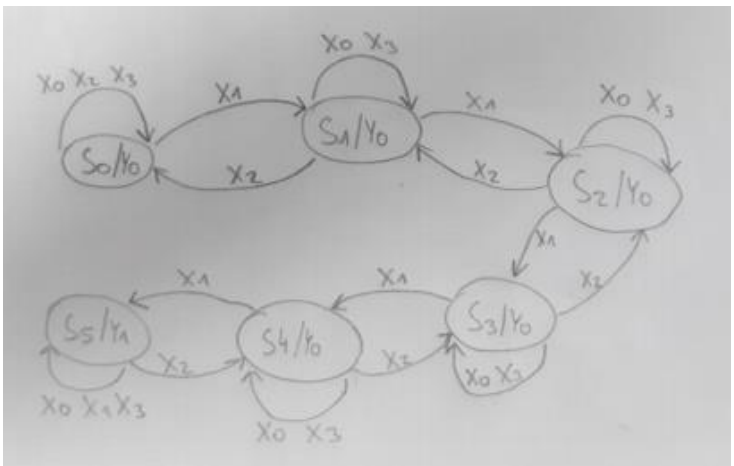
vstupní	vnitřní	výstupní
p o	stavy	semafor
X0 0 0	S0 000	y0 0
X1 1 0	S1 001	y1 1
X2 0 1	S2 010	
X3 1 1	S3 011	
	S4 100	
	S5 101	

4) Popis automatu Moore nebo Mealy

Mealyho automat – 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.

Mooreův automat – 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) Orientovaný 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 – programy

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;


```

Dekodér

```
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;


```

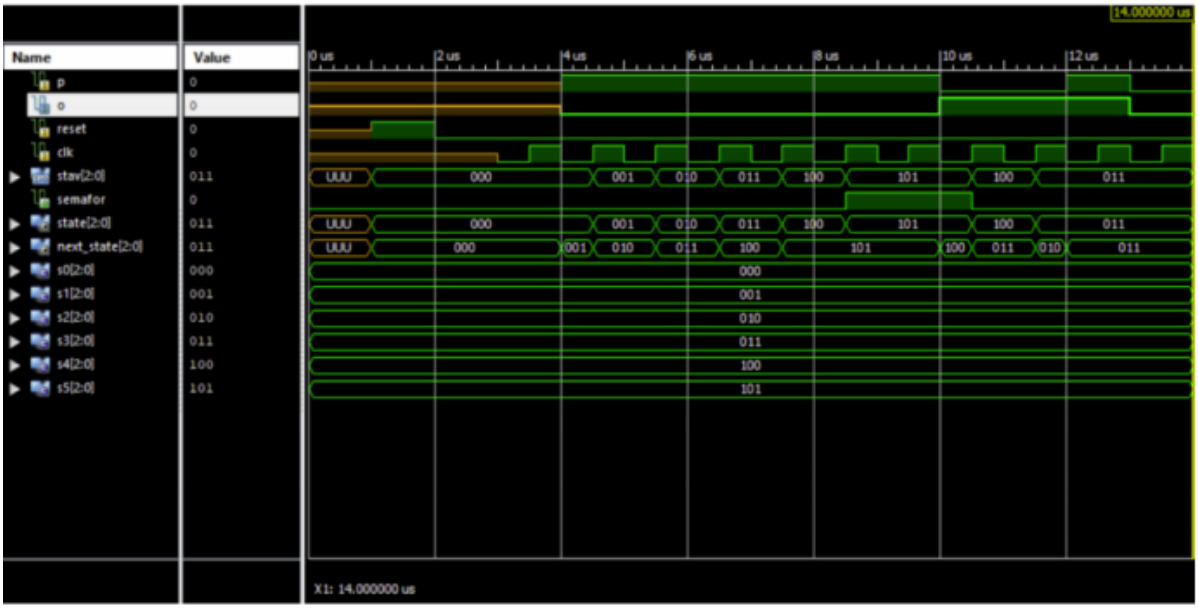
Garáž – main

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


entity garaz_main 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_main;

architecture Behavioral of garaz_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";
   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;

 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;
```

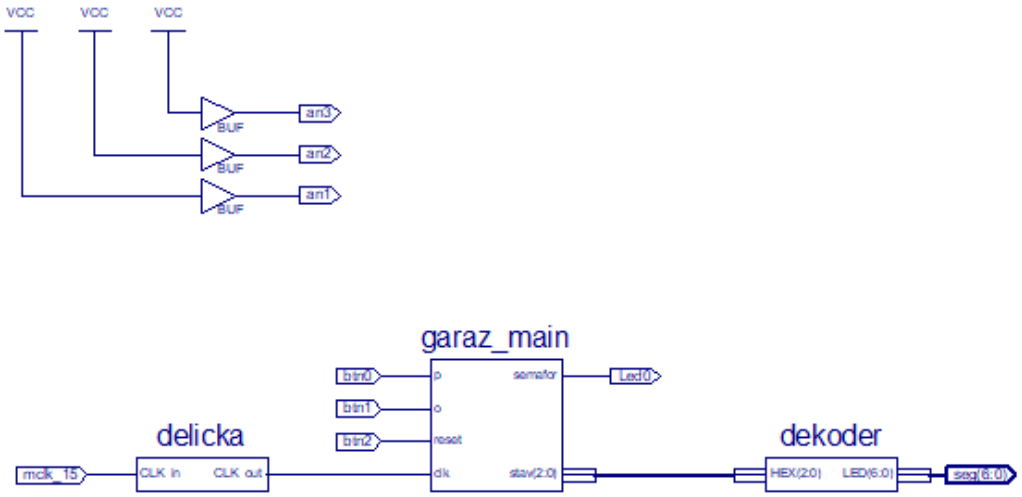
```
55      else
56        next_state <= S0;
57      end if;
58
59
60    when S1 =>
61      if (p = '1' AND o = '0') then
62        next_state <= S2;
63      elsif (p = '0' AND o = '1') then
64        next_state <= S0;
65      else
66        next_state <= S1;
67      end if;
68
69
70    when S2 =>
71      if (p = '1' AND o = '0') then
72        next_state <= S3;
73      elsif (p = '0' AND o = '1') then
74        next_state <= S1;
75      else
76        next_state <= S2;
77      end if;
78
79
80    when S3 =>
81      if (p = '1' AND o = '0') then
82        next_state <= S4;
83      elsif (p = '0' AND o = '1') then
84        next_state <= S2;
85      else
86        next_state <= S3;
87      end if;
88
89
90    when S4 =>
91      if (p = '1' AND o = '0') then
92        next_state <= S5;
93      elsif (p = '0' AND o = '1') then
94        next_state <= S3;
95      else
96        next_state <= S4;
97      end if;
98
99
100   when S5 =>
101     if (p = '0' AND o = '1') then
102       next_state <= S4;
103     else
104       next_state <= S5;
105     end if;
106
107    when others => NULL;
108     end case;
```

```
109
110     stav <= state;
111     end process ZAKODOVANI_STAVU;
112
113   end Behavioral;
114
115
```

8) Simulace



9) Celkové schéma



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

Zpočátku se mi nepodařilo spustit simulaci, ale chyba byla ve špatně připojeném pinu u schémata. Projekt bych vylepšil LED diodou, která by vždy problikla při příjezdu nebo odjezdu.

Tomáš Konvičný 28.10.2020