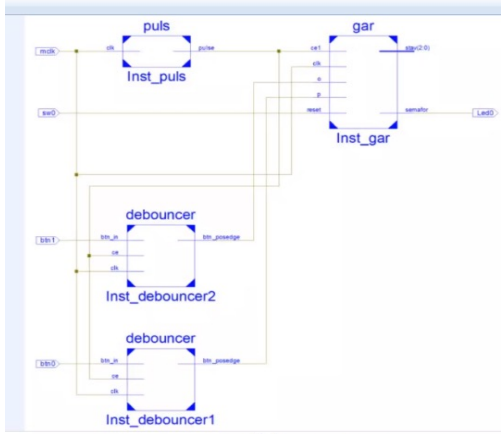


Střední průmyslová škola elektrotechnická Haviřov	Třída: 4.B
	Skupina: 1
Mooreův automat	Zpráva číslo: 2
	Den: 15.2.2021
 <p>The diagram shows a Moore machine implementation using logic blocks. It includes a 'puls' block (Inst_puls) with inputs 'clk' and 'res', and outputs 'set' and 'clr'. A 'gar' block (Inst_gar) has inputs 'set', 'clr', 'n', 'p', 'res', and 'sample', and outputs 'agg(0-3)' and 'sample'. Two 'debouncer' blocks (Inst_debouncer1 and Inst_debouncer2) have inputs 'clk', 'set', and 'clr', and outputs 'set' and 'clr'. The diagram also shows a 'Led0' output. The circuit is connected to a 'clk' input and a 'res' input. The 'set' and 'clr' outputs of the 'puls' block are connected to the 'set' and 'clr' inputs of the 'gar' block. The 'set' and 'clr' outputs of the 'debouncer' blocks are connected to the 'set' and 'clr' inputs of the 'gar' block. The 'sample' output of the 'gar' block is connected to the 'Led0' output.</p>	
	Jméno učitele:
	Ing. Božena Ralbovská
	Jméno: Jakub Lengsfeld
	Známka:

## 1. Zadání - měření intenzity světla

Pomocí FPGA obvodu Spartan 3E firmy Xilinx realizujte automat pro řízení chodu **garáže pro 5 automobilů**.

Automat bude signalizovat stavy volno a obsazeno podle příjezdějících a odjíždějících aut. Příjezd a odjezd nastavujte tlačítky, nulování pomocí přepínače, počet aut zobrazujte na sedmi segmentovém displeji. Ošetřete zákmity tlačítek proti chybnému počítání aut.

## 2. Teoretický rozbor

### Spartan:

-Je programovatelné hradlové pole, jedná se o typ integrovaného obvodu).

-Jedná se účelově programovatelné zařízení, pracuje pomocí instrukcí uložených v paměti. Přijatá data zpracovává a posílá na výstupy.

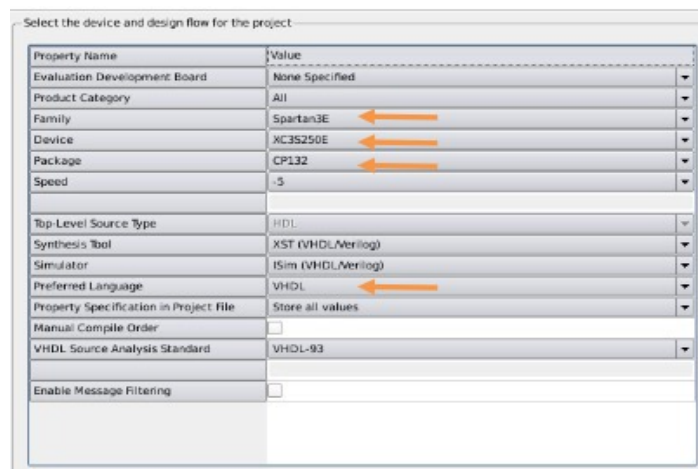
### VHDL:

-Jedná se o programovací jazyk. Používá se pro návrh simulací digitálních integrovaných obvodů.

-Například programovatelných hradlových polí nebo různých zákaznických obvodů.

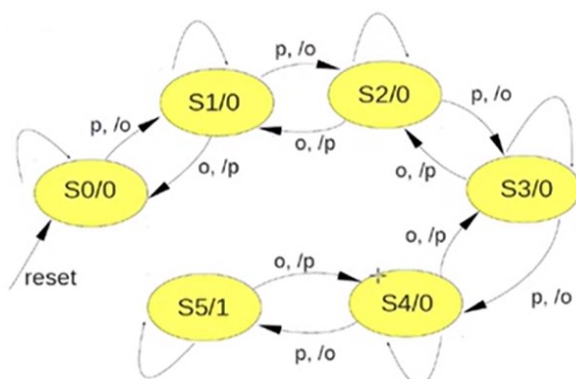
-Umožňuje návrh jak logických, tak i sekvenčních struktur.

### Nastavení VHDL:



**Mooreův automat** - Změna na vstupu se u něj projeví na výstupu až v následujícím stavu

- funkce jsou tedy funkcemi pouze vnitřního stavu



Stav	kod
S0	000
S1	001
S2	010
S3	011
S4	100
S5	101

Stav	Semafor
S0	0
S1	0
S2	0
S3	0
S4	0
S5	1

### 3.1 Program

```
entity garaz is
    Port ( semafor : out  STD_LOGIC;
          stav : out  STD_LOGIC_VECTOR (2 downto 0);
          o : in  STD_LOGIC;
          p : in  STD_LOGIC;
          reset : in  STD_LOGIC;
          ce1,clock |: in  STD_LOGIC);
end garaz;

architecture Behavioral of garaz is

    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";

    signal state, next_state : STD_LOGIC_VECTOR(2 downto 0);
```

p - příjezd auta

o - odjezd auta

reset - vrátí stav do s0

semafor - když je plná garáž svítí červená

ce1 - chip enable

clock - hodinový signál

stav - kolik aut je v garáži

```
begin

    SYNC_PROC: process (clock)
    begin
        if (clock'event and clock = '1') then
            if (reset = '1') then
                state <= s0;
            else
                state <= next_state;
            end if;
        end if;
    end process;
```

když je náběžná hrana a zároveň reset je v log 1 stav se nastaví na s0

pokud ne nastaví se další stav

nastavení semaforu, svítí jen když stav je s5 (plná garáž)

```
OUTPUT_DECODE: process (state)
begin
    if state = s5 then
        semafor <= '1';
    else
        semafor <= '0';
    end if;
end process;
```

```

NEXT_STATE_DECODE: process (state,p,o)
begin

    next_state <= state;

    case (state) is

        when s0 => if p= '1' and o='0' then next_state <=s1;
                    else next_state <=state;
                    end if;

        when s1 => if p = '1' and o='0' then next_state <=s2;
                    elsif p = '0' and o='1' then next_state <= s1;
                    else next_state <=state;
                    end if;

        when s2 => if p = '1' and o='0' then next_state <=s3;
                    elsif p = '0' and o='1' then next_state <= s2;
                    else next_state <=state;
                    end if;

        when s3 => if p = '1' and o='0' then next_state <=s4;
                    elsif p = '0' and o='1' then next_state <= s3;
                    else next_state <=state;
                    end if;

        when s4 => if p = '1' and o='0' then next_state <=s5;
                    elsif p = '0' and o='1' then next_state <= s4;
                    else next_state <=state;
                    end if;

        when s5 => if p = '0' and o='1' then next_state <= s4;
                    else next_state <=state;
                    end if;

        when others => null;

    end case;
    stav<=state;
end process;

```

nastavení dalších stavů, když je stav s0 - s4 tak platí : pokud někdo přijede a nikdo neodjede, tak se posuneme o stav nahoru

pokud nikdo nepřijede a někdo odjede posuneme se o stav dolů

výjimka je jen při stavu s5 kdy se vynechá první podmínka a rozsvítí se semafor

```

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

entity top_gar is
    Port ( mclk : in  STD_LOGIC;
          btn0 : in  STD_LOGIC;
          btn1 : in  STD_LOGIC;
          sw0 : in  STD_LOGIC;
          led0 : out STD_LOGIC;
          seg : out  STD_LOGIC_VECTOR (6 downto 0);
          an : out  STD_LOGIC_VECTOR (3 downto 0));
end top_gar;

architecture struc of top_gar is
    signal cel,p,o:STD_LOGIC;
    signal cislo:STD_LOGIC_VECTOR (2 downto 0);
begin

    Inst_puls: entity puls PORT MAP(
        clk => mclk,
        pulse => cel
    );

    Inst_debouncer1: entity debouncer PORT MAP(
        clk => mclk,
        ce => cel,
        btn_in => btn0,
        btn_posedge => p
    );

    Inst_debouncer2: entity debouncer PORT MAP(
        clk => mclk,
        ce => cel,
        btn_in => btn1,
        btn_posedge => o
    );

    Inst_gar: entity gar PORT MAP(
        clk => mclk,
        reset => sw0,
        p => p,
        o => o,
        cel => cel,
        semafor => led0,
        stav => cislo
    );

    with cislo SELECT
        seg<= "1111001" when "001",  --1
              "0100100" when "010",  --2
              "0110000" when "011",  --3
              "0011001" when "100",  --4
              "0010010" when "101",  --5
              "1000000" when others;  --0
    an(3 downto 0) <= "1011";

end struc;

```

## 3.2 Piny

```

2  # Connected to Basys2 onBoard 7seg display
3  NET "seg<0>" LOC = "L14"; # Bank = 1, Signal name = CA
4  NET "seg<1>" LOC = "H12"; # Bank = 1, Signal name = CB
5  NET "seg<2>" LOC = "N14"; # Bank = 1, Signal name = CC
6  NET "seg<3>" LOC = "N11"; # Bank = 2, Signal name = CD
7  NET "seg<4>" LOC = "P12"; # Bank = 2, Signal name = CE
8  NET "seg<5>" LOC = "L13"; # Bank = 1, Signal name = CF
9  NET "seg<6>" LOC = "M12"; # Bank = 1, Signal name = CG
0  ##NET "dp" LOC = "N13"; # Bank = 1, Signal name = DP
1  #
2  NET "an<3>" LOC = "K14"; # Bank = 1, Signal name = AN3
3  NET "an<2>" LOC = "M13"; # Bank = 1, Signal name = AN2
4  NET "an<1>" LOC = "J12"; # Bank = 1, Signal name = AN1
5  NET "an<0>" LOC = "F12"; # Bank = 1, Signal name = AN0
6

```

## 4. Simulace



## **5. Zhodnocení**

Jednoduchý program na plnění garáže. Pro takhle malý počet jako je 5 míst pro auta dostačující, avšak kdybychom museli napsat program pro 100 míst použili by jsme čítač nebo nějaké jiné řešení.

Jediný problém mi dělalo testování funkčnosti, z důvodu nedostupnosti spartana.