Střední průmyslová škola elektrotechnická Havířov	**PRG**	**Třída: 4B** **Skupina:** 1

Posuvný registr

Den: 02.04.2025

Jméno učitele: Božena Ralbovská

Jméno: Vojtěch Lisztwan

Známka: snad jedna

1. Zadání

Pomocí FPGA obvodu Spartan 3E firmy Xillinx realizujte automat pro řízení kruhového registru. Vzorek dat pro kruhový registr nastavte na osmi přepínačích. Prvním tlačítkem načtěte nový vzorek dat. Obsah registru zobrazte na 8 LED diod. Druhé tlačítko přepíná směr rotace, použijte automat se dvěma stavy(zap.,vyp.) .Tlačítko reset nám zhasne LED diody.

2. Teoretický rozbor - analýza

Mealyho vs Mooreův automat

Mooreův automat

- V Mooreově automatu závisí výstup pouze na aktuálním stavu.
- Každý stav má pevně definovaný výstup.
- Výstup se mění jen při změně stavu, ne při změně vstupu v rámci jednoho stavu.
- Je stabilnější a jednodušší na návrh.

Mealyho automat

- V Mealyho automatu závisí výstup jak na aktuálním stavu, tak na vstupu.
- Výstup se může okamžitě měnit při změně vstupu, i když stav zůstane stejný.
- Může být rychlejší, protože reaguje hned na změnu vstupu.
- Návrh je složitější než u Mooreova automatu.

Struktura kódu

Soubor had.vhd

- Řídí výstup led na základě směru, resetu a loadu.
- Posouvání registru o 1 bit.

Soubor puls.vhd

- Zajišťuje časování(dělička frekvence clk)

Soubor debouncer.vhd

- Zajišťuje správnou funkčnost tlačítka pro load.
- Odstraňuje zákmity z tlačítka

Soubor sta.vhd

- Soubor, který obsahuje MOOREŮV automat.
- Na základě tlačítka mění stavy (zap a vyp)

- Výstup této instance představuje směr rotování ledek.

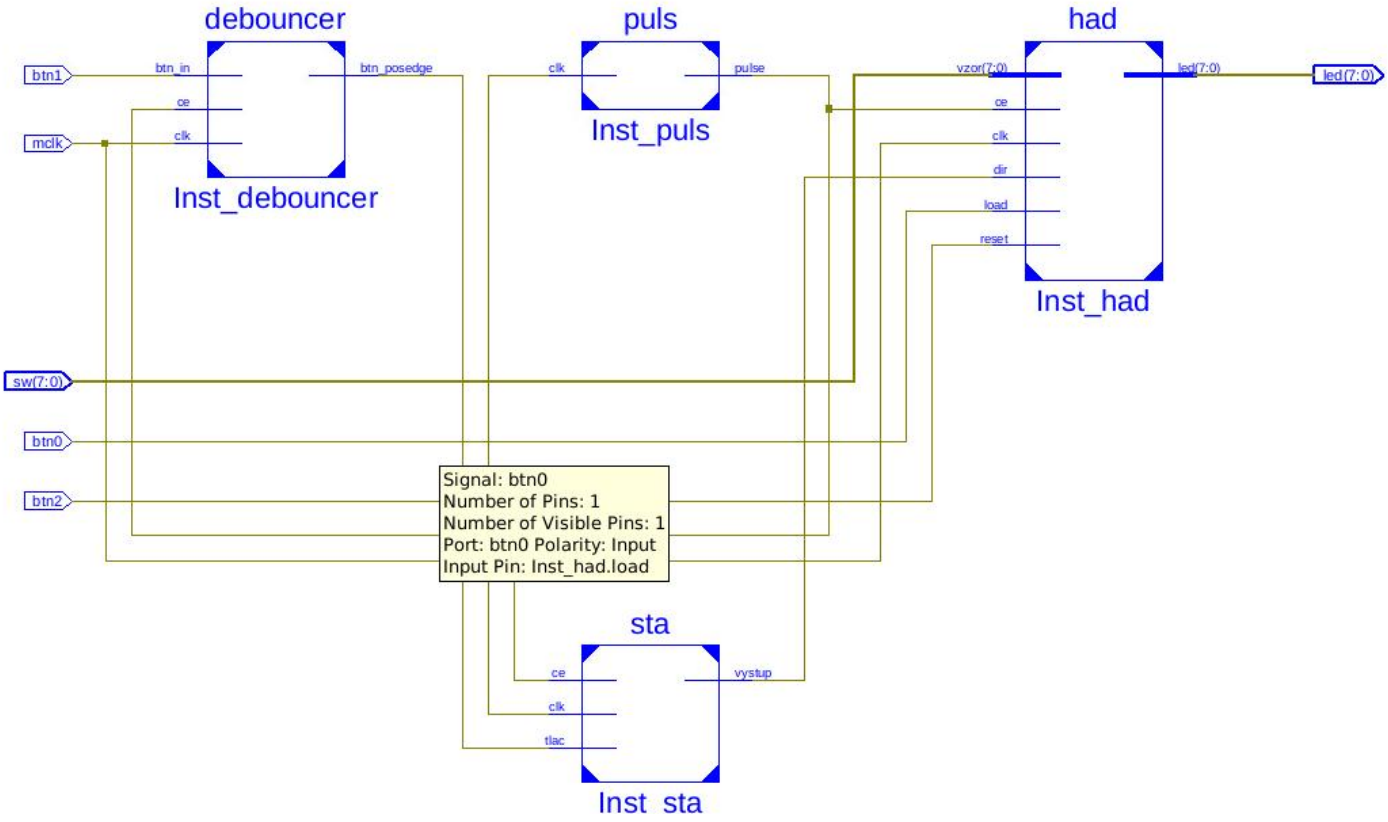
Soubor top_had1.vhd

- Soubor, který propojuje jednotlivé instance do jednoho funčního celku.

Soubor piny_spartan.ucf

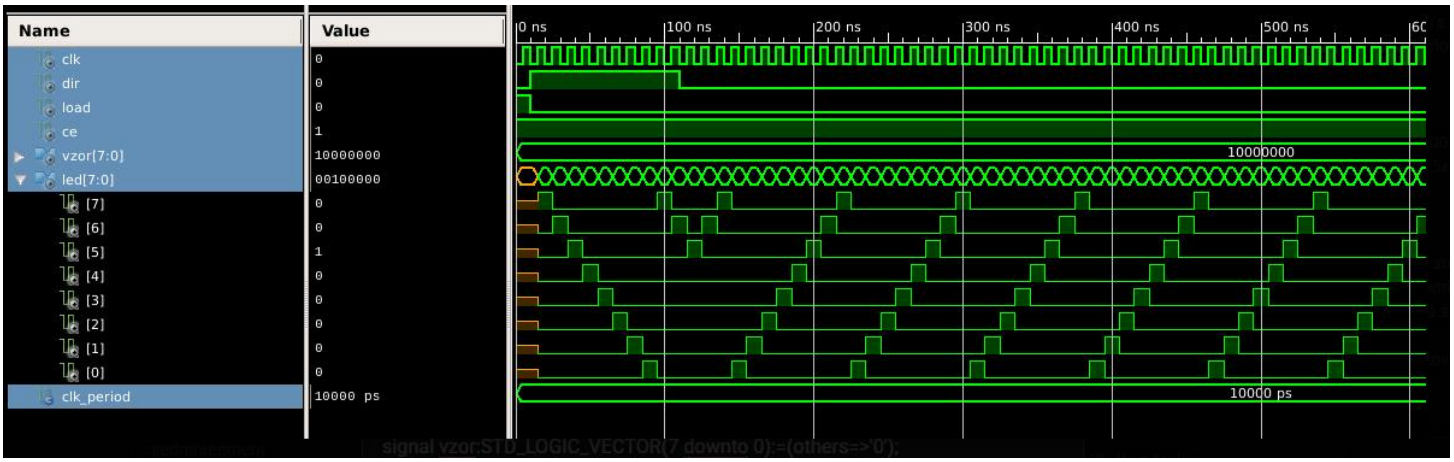
- Přiřazení jednotlivých hardwarových komponent k proměnným programu.

Propojení jednotlivých entit

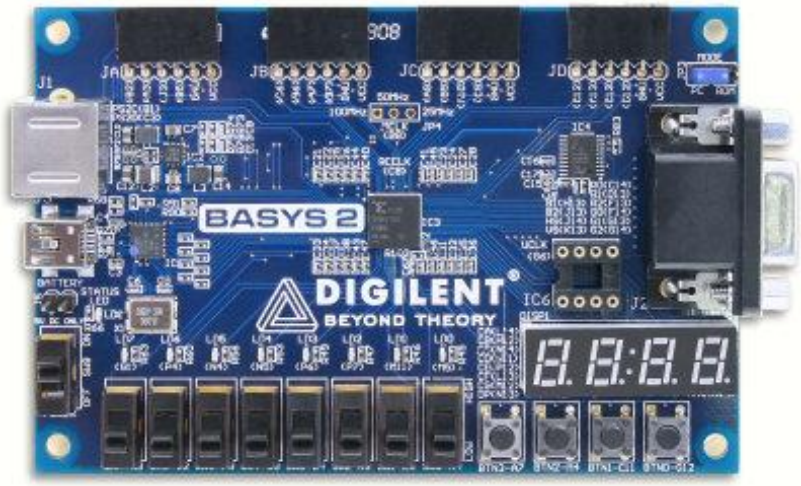


Simulace

- Ověření funkčnosti souboru had.vhd v simulaci



Hardware

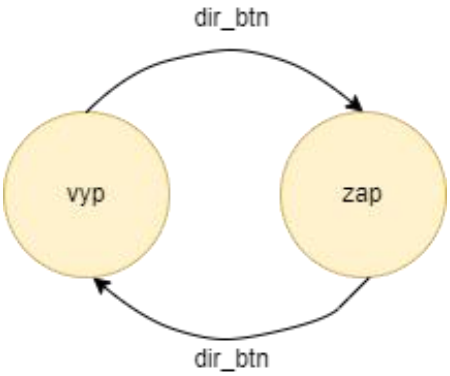


Deska Digilent Basys 2 je vývojová deska s FPGA čipem Xilinx Spartan-3E, která je určena pro výuku a prototypování digitálních systémů.

Hlavní vlastnosti desky:

- FPGA čip: Xilinx Spartan-3E (XC3S250E nebo XC3S100E, podle verze)
- Paměť: 16 MB flash paměti (SPI)
- Taktovací oscilátor: 50 MHz
- Vstupy a výstupy:
  - 8 LED diod
  - 4 tlačítka
  - 8 přepínačů
  - 4místný 7segmentový displej
  - PS/2 konektor pro klávesnici/myš
  - VGA výstup pro zobrazování grafiky

3. Stavový diagram



4. Program

Soubor had.vhd

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


use IEEE.NUMERIC_STD.ALL;


entity had is
    Port ( clk : in  STD_LOGIC;
           dir : in  STD_LOGIC;
           load : in  STD_LOGIC;
              reset : in  STD_LOGIC;
           ce : in  STD_LOGIC;
           vzor : in  STD_LOGIC_VECTOR (7 downto 0);
           led : out  STD_LOGIC_VECTOR (7 downto 0));
end had;

architecture Behavioral of had is
signal temp_reg:unsigned (7 downto 0);

begin

process (clk)
   begin
      if (clk'event and clk = '1') then
         if (ce = '1') then
                if(reset='1')then
                    temp_reg<="00000000";
                    led<= STD_LOGIC_VECTOR(temp_reg);
                else
```

```
            if(load='1')then
                    temp_reg<=unsigned(vzor);
                else
                    if(dir='1')then
                        temp_reg <= temp_reg ror 1;
                    else
                        temp_reg <= temp_reg rol 1;
                    end if;
                    led<= STD_LOGIC_VECTOR(temp_reg);
                end if;
                end if;
            end if;
        end if;
    end process;


end Behavioral;
```

Soubor puls.vhd

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

entity puls is
  generic (
      div_factor  : integer := 5000000);  -- 100 ms
  port(
      clk   : in std_logic;
      pulse : out std_logic
  );
end puls;

Architecture Behavioral of puls is
signal counter : integer := 0;

begin
process(clk)
  begin
    if(rising_edge(clk)) then
        if counter = div_factor - 1 then
            pulse <= '1';
            counter <= 0;
        else
            counter <= counter + 1;
            pulse <= '0';
        end if;
    end if;
end process;
end Behavioral;
```

Soubor debouncer.vhd

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
```

```
entity debouncer is
    port (
        clk : in std_logic;
        ce  : in std_logic;
        btn_in      : in  std_logic;
        btn_posedge : out std_logic
        );

end entity debouncer;

architecture str of debouncer is

    signal debounce_reg : std_logic_vector(15 downto 0);

begin

    deb_proc : process (clk) is
    begin
        if rising_edge(clk) then

                -- make sure pulse is 1clk wide
                --btn_posedge <= '0';

                if ce = '1' then
                    debounce_reg <= debounce_reg(debounce_reg'high -1 downto 0) & btn_in;

                        if debounce_reg = X"00FF" then
                              btn_posedge <= '1';

                        else btn_posedge <= '0';
                        end if;
            end if;
        end if;
    end process deb_proc;

end architecture str;
```

Soubor sta.vhd

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

entity sta is
    Port ( clk : in  STD_LOGIC;
           tlac : in  STD_LOGIC;
           ce : in  STD_LOGIC;
           vystup : out  STD_LOGIC);
end sta;

architecture Behavioral of sta is
type state_type is(zap, vyp);
    signal state, nextstate : state_type;

begin
```

```
SYNC_PROC: process (clk)
   begin
      if (clk'event and clk = '1') then
         if (ce = '1') then
            state <= nextstate;
         end if;
      end if;
   end process;


OUTPUT_DECODE: process (state)
   begin

      if state = zap then
         vystup <= '1';
      else
         vystup <= '0';
      end if;
   end process;

NEXT_STATE_DECODE: process (state, tlac)
   begin
      nextstate <= state;

      case (state) is
         when zap =>
            if tlac = '1' then
               nextstate <= vyp;
            end if;
         when vyp =>
            if tlac = '1' then
               nextstate <= zap;
            end if;

         when others =>
           null;
      end case;
   end process;


end Behavioral;
```

Soubor top_had1.vhd

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


entity top_had1 is
    Port ( mclk : in  STD_LOGIC;
           btn0 : in  STD_LOGIC;
           btn1 : in  STD_LOGIC;
              btn2 : in  STD_LOGIC;
           sw : in  STD_LOGIC_VECTOR (7 downto 0);
```

```
           led : out  STD_LOGIC_VECTOR (7 downto 0));
end top_had1;

architecture Behavioral of top_had1 is
signal ce, dir, pos_edge:STD_LOGIC;
signal vzor:STD_LOGIC_VECTOR(7 downto 0):=(others=>'0');

begin
Inst_had:entity had PORT MAP(
    clk =>mclk ,
    dir => dir,
    load => btn0,
    reset => btn2,
    ce => ce,
    vzor => sw,
    led =>led
);

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

Inst_debouncer:entity debouncer PORT MAP(
    clk => mclk,
    ce => ce,
    btn_in => btn1,
    btn_posedge =>pos_edge
);

Inst_sta:entity sta PORT MAP(
    clk => mclk,
    tlac => pos_edge,
    ce => ce,
    vystup =>dir
);

end Behavioral;
```

Soubor piny_spartan.ucf

```
# 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 "led(0)" LOC = "M5" ;  # Bank = 2, Signal name = LD0


# Pin assignment for SWs

NET "sw(7)" LOC = "N3";  # Bank = 2, Signal name = SW7

NET "sw(6)" LOC = "E2";  # Bank = 3, Signal name = SW6

NET "sw(5)" LOC = "F3";  # Bank = 3, Signal name = SW5

NET "sw(4)" LOC = "G3";  # Bank = 3, Signal name = SW4

NET "sw(3)" LOC = "B4";  # Bank = 3, Signal name = SW3

NET "sw(2)" LOC = "K3";  # Bank = 3, Signal name = SW2

NET "sw(1)" LOC = "L3";  # Bank = 3, Signal name = SW1

NET "sw(0)" 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
```

5. Zhodnocení

Projekt je dobře navržený a plně funkční. Má jasnou strukturu a je snadno rozšiřitelný. Některá vylepšení by mohla přidat více možností ovládání a zpřehlednit kód, ale i v současné podobě jde o povedenou práci.

- Modulární uspořádání: Kód je přehledně rozdělen do samostatných bloků, což usnadňuje práci a případné úpravy.
- Debouncer tlačítek: Pomáhá odstranit rušení a zajistit spolehlivé rozpoznání stisknutí.
- Generátor pulzů: Zajišťuje pravidelné přepínání stavů.
- Stavový automat: Umožňuje řídit směr posunu LED, což je jednoduché a efektivní.
- Logické propojení: Signály jsou jasně definované a správně propojené mezi moduly.