Układy cyfrowe i systemy wbudowane - laboratorium

Karol Kulawiec 241281
Bartosz Rudnikowicz 241382

16.12.2019

1 Wstęp

Naszym celem była modyfikacja projektu zamka szyfrowego, który został wykonany na poprzednich laboratoriach w celu uruchomienia go na układzie FPGA, a następnie dodanie obsługi wyświetlacza LCD, oraz stworzenie test-bencha.

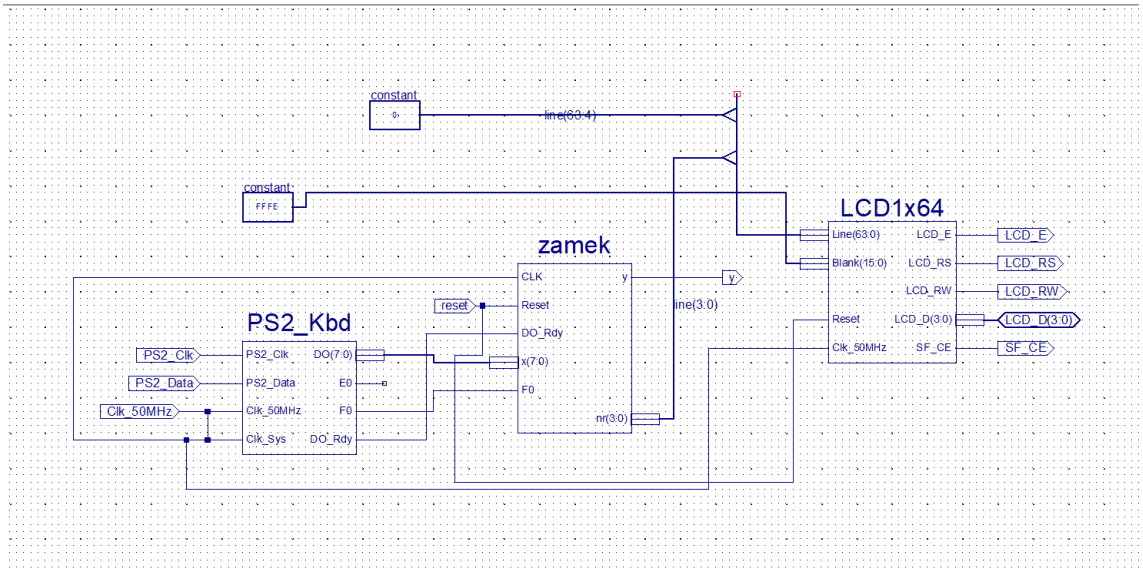
2 Przebieg zajęć

2.1 Uruchomienie poprzedniego modułu na nowym sprzęcie

Pierwszym zadaniem "rozgrzewkowym" było wykorzystanie poprzedniego projektu i przerobieniu go tak, aby dział na układzie Spartan 3E. Wymagało to między innymi zmienienia modułów dla układu CPLD na odpowiadające im moduły dla układu Spartan.
Po dokonaniu wszystkich niezbędnych modyfikacji, udało nam się uruchomić poprawnie działający układ, który bez problemu wykrywał zadany ciąg "ABCD".

2.2 Rozbudowa projektu o ekran LCD i moduł PS2_Kbd

Kolejnym zadaniem było użycie modułu PS2_Kbd, który ma sygnał wyjściowy F0 informujący czy dany kod klawisza został poprzedzony tym właśnie kodem, czyli czy oznaczał zwolnienie tego klawisza. Dzięki temu mogliśmy przebudować nasz zamek na maszynę pięcio-stanową. Stan maszyny zmieniał się w chwili naciśnięcia danego klawisza (a nie dopiero po jego zwolnieniu).
Dodatkowo użyliśmy modułu LCD1x64, który wyświetlał numer aktualnego stanu maszyny.
Poniżej znajduje się kod oraz schemat projektu (Rysunek 1).



Rysunek 1: Schemat projektu zamka szyfrowego.

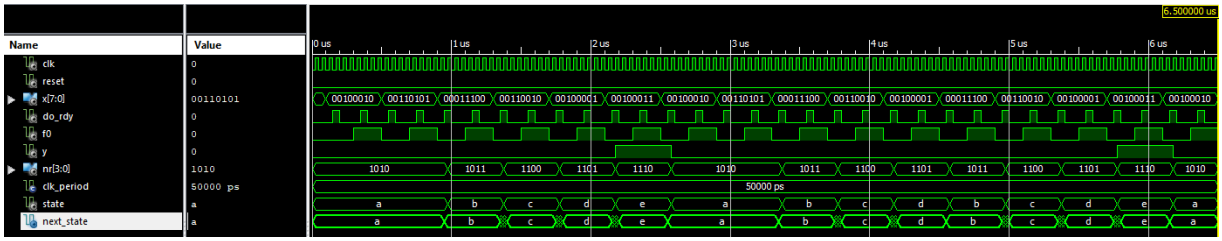
```
entity zamek is
    Port ( CLK : in  STD_LOGIC;
           Reset : in  STD_LOGIC;
           x : in  STD_LOGIC_VECTOR (7 downto 0);
           y : out  STD_LOGIC;
              nr: out STD_LOGIC_VECTOR (3 downto 0);
           DO_Rdy : in  STD_LOGIC;
              F0 : in STD_LOGIC
              );
end zamek;

architecture Behavioral of zamek is

    type state_type is ( A, B, C, D, E);
    signal state, next_state : state_type;

begin

    process1 : process(CLK)
    begin
        if rising_edge(CLK) then
            if Reset = '1' then
                state <= A;
            else
                state <= next_state;
            end if;
        end if;
    end process process1;

    process2: process(Do_Rdy, state, X)
    begin
    next_state <= state;
    if Do_Rdy = '1' and F0 = '0' then
        case state is
        when A => if x = X"1C" then next_state <= B; end if;
        when B => if x = X"32" then next_state <= C; elsif x = X"1C"
            then next_state <= B; else next_state <= A; end if;
        when C => if x = X"21" then next_state <= D; elsif x = X"1C"
            then next_state <= B; else next_state <= A; end if;
        when D => if x = X"23" then next_state <= E; elsif x = X"1C"
            then next_state <= B; else next_state <= A; end if;
        when E => if x = X"1C" then next_state <= B; else next_state <=
            A; end if;
        end case;
    end if;
end process process2;

y <= '1' when state = E else '0';

with state select
    nr<= X"A" when A,
    X"B" when B,
    X"C" when C,
    X"D" when D,
    X"E" when E,
    X"A" when others;

end Behavioral;
```

2.3 Symulacja układu

Ostatnim zadaniem było przygotowanie test bencha dla stworzonego układu.
Na wejściu podany został ciąg znaków "XYABCDXYABCABCDXY" i dla niego została przeprowadzona symulacja. Poniżej znajduje się napisany przez nas kod oraz wynik symulacji (Rysunek 2. Jak widać układ poprawnie wykrył oba ciągi "ABCD" i nie pomylił się dla ciągu "ABC".

```
ARCHITECTURE behavior OF zad1tb IS

    -- Component Declaration for the Unit Under Test (UUT)

    COMPONENT zamek
    PORT(
         CLK : IN  std_logic;
         Reset : IN  std_logic;
         x : IN  std_logic_vector(7 downto 0);
         y : OUT  std_logic;
         nr : OUT  std_logic_vector(3 downto 0);
         DO_Rdy : IN  std_logic;
         F0 : IN  std_logic
        );
    END COMPONENT;


   --Inputs
    signal CLK : std_logic := '0';
    signal Reset : std_logic := '0';
    signal x : std_logic_vector(7 downto 0) := (others => '0');
    signal DO_Rdy : std_logic := '0';
    signal F0 : std_logic := '0';

    --Outputs
    signal y : std_logic;
    signal nr : std_logic_vector(3 downto 0);

   -- Clock period definitions
    constant CLK_period : time := 50 ns;

BEGIN

    -- Instantiate the Unit Under Test (UUT)
    uut: zamek PORT MAP (
           CLK => CLK,
           Reset => Reset,
           x => x,
           y => y,
           nr => nr,
           DO_Rdy => DO_Rdy,
           F0 => F0
         );

   -- Clock process definitions
    CLK_process :process
    begin
         CLK <= '0';
         wait for CLK_period/2;
         CLK <= '1';
         wait for CLK_period/2;
    end process;


   -- Stimulus process
```

```
    stim_proc: process

     type typeByteArray is array (NATURAL range <> ) of STD_LOGIC_VECTOR
          ( 7 downto 0);
     variable arrBytes : typeByteArray(0 to 16)
          := (X"22", X"35", X"1C", X"32", X"21", X"23", X"22", X"35", X"1
              C", X"32", X"21", X"1C", X"32", X"21", X"23", X"22", X"35")
              ;

    begin
       wait for 100 ns;

          for i in arrBytes'RANGE loop
              F0 <= '0';



              x <= arrBytes(i);
              wait for CLK_period;
              DO_Rdy <= '1';
              wait for CLK_period;
              DO_Rdy <= '0';
              wait for 2*CLK_period;


              F0 <= '1';
              wait for CLK_period;
              DO_Rdy <= '1';
              wait for CLK_period;
              DO_Rdy <= '0';
              wait for 2*CLK_period;


          end loop;

       wait;
    end process;

END;

end zamek;
```



Rysunek 2: Wynik symulacji.

3 Wnioski

Podczas zajęć zaprojektowaliśmy, zaimplementowaliśmy oraz zasymulowaliśmy zadany układ. Podczas wykonywania zadań nie napotkaliśmy większych problemów, a układ zadziałał prawidłowo.