Technika Cyfrowa

Sprawozdanie z ćwiczenia nr 4

Natalia Curzytek, Marta Stanisławska,
Karolina Nitsch, Szymon Kłodowski

Treść zadania

Korzystając z programu Quartus firmy Altera (Intel) (www.altera.com), należy w dowolnym wybranym układzie scalonym FPGA stworzyć działający system sterujący, realizujący bardzo prosty wybór z menu. Mamy do dyspozycji dwa wyświetlacze siedmiosegmentowe (lewy i prawy) oraz dwa przyciski (lewy i prawy). Wciśnięcie lewego przycisku powoduje zwiększenie o jeden wartości wyświetlanej na lewym wyświetlaczu. Po osiągnięciu wartości 9, wartość ta zmienia się na 0 przy ponownym wciśnięciu lewego przycisku. Po już ustawieniu lewym przyciskiem żądanej wartości na lewym wyświetlaczu, wciśnięcie prawego przycisku powinno spowodować zapamiętanie wartości z lewego wyświetlacza na wyświetlaczu prawym, co świadczy o dokonanym wyborze wartości z menu.

Po uruchomieniu systemu, obydwa wyświetlacze powinny pokazywać wartość zero. Całość powinna działać zgodnie z filmem z poniższego linku:

fpga-menu.mp4

Czym są układy FPGA

Układy FPGA (Field-Programmable Gate Array) to programowalne matryce bramek logicznych, które pozwalają na tworzenie specyficznych konfiguracji sprzętowych poprzez odpowiednie zaprogramowanie ich struktury. W przeciwieństwie do tradycyjnych układów scalonych, które mają stałą funkcjonalność, FPGA mogą być dowolnie konfigurowane. Dzięki temu znajdują szerokie zastosowanie w wielu dziedzinach technologii.

Wnętrze FPGA składa się z dużej liczby powtarzalnych bloków logicznych (LAB - Logic Array Block), wbudowanych bloków tablicowych (EAB - Embedded Array Block) oraz elementów połączeniowych - tzw. magistrali połączeń. Podstawowym elementem funkcjonalnym FPGA jest LUT (Look-Up Table) – tablica, która implementuje funkcje logiczne. LUT może odwzorowywać dowolną funkcję logiczną zależną od kilku wejść, co czyni go bardzo uniwersalnym blokiem budulcowym.

Jedną z kluczowych cech FPGA jest równoległość. W przeciwieństwie do klasycznych procesorów (CPU), które wykonują instrukcje sekwencyjnie, FPGA pozwala na jednoczesne (równoległe) działanie wielu bloków logicznych. Oznacza to, że projektując układ w FPGA, można zrealizować wiele operacji w tym samym czasie, co przekłada się na bardzo wysoką wydajność.

Zastosowania praktyczne układów FPGA

Układy FPGA znajdują szerokie zastosowanie w nowoczesnych systemach cyfrowych dzięki możliwości elastycznej konfiguracji oraz równoległego przetwarzania danych. Poniżej przedstawiono różne sposoby ich wykorzystania:

- **Integracja FPGA z mikroprocesorami**
  Układy FPGA mogą współpracować z mikroprocesorami w ramach kart rozszerzających do komputerów PC.
- **FPGA z wbudowanymi mikroprocesorami**
  Nowoczesne układy FPGA mogą zawierać w sobie zintegrowane rdzenie mikroprocesorowe, umożliwiając uruchamianie systemów operacyjnych bez potrzeby stosowania zewnętrznych komponentów. Przykładem mogą być układy SmartFusion, które łączą w sobie FPGA, rdzeń procesora oraz elementy analogowe, co pozwala tworzyć kompleksowe systemy wbudowane.
- **FPGA z wbudowanym mikroprocesorem programowalnym (soft-core)**
  Istnieje możliwość implementacji tzw. miękkiego procesora (soft-core) w strukturze FPGA. Tego typu rozwiązanie umożliwia programową konfigurację parametrów procesora, takich jak zestaw instrukcji, rozmiar rejestrów czy sposób obsługi przerwań, co pozwala na dostosowanie jednostki obliczeniowej do konkretnego zastosowania.
- **Zastosowanie wyłącznie FPGA**
  W wielu przypadkach FPGA może działać samodzielnie, bez wsparcia dodatkowego procesora. Przykładowo, można zrealizować system monitorowania środowiska, integrując w FPGA obsługę wielu czujników i przetwarzanie danych w czasie rzeczywistym.

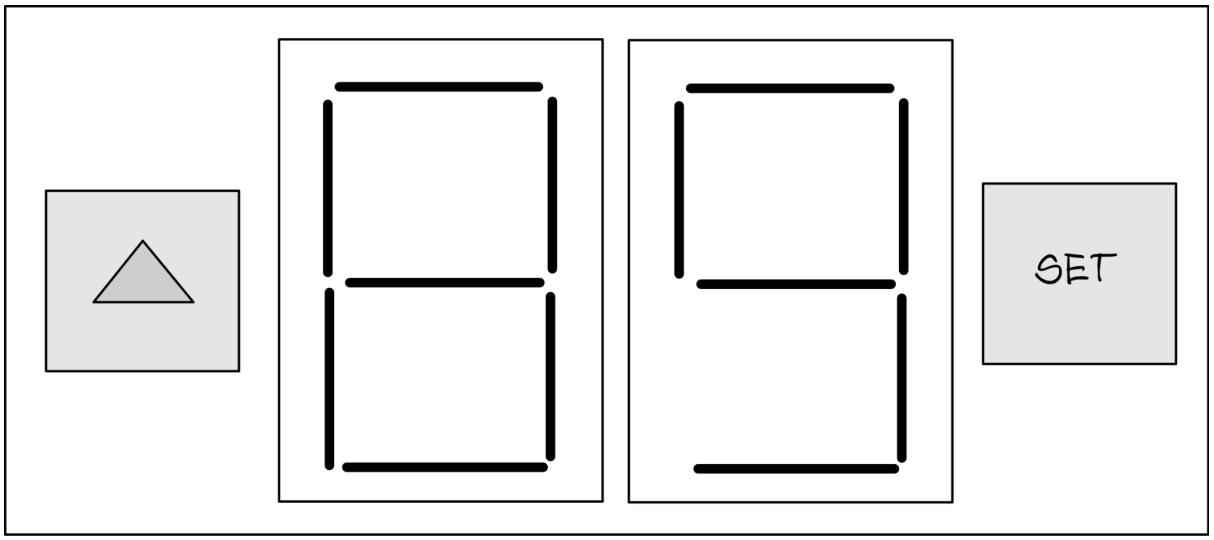
Układy FPGA odgrywają istotną rolę w informatyce i nauce, umożliwiając realizację zadań wymagających dużej wydajności i równoległego przetwarzania danych. Przykładowe zastosowania obejmują:

- **Przyspieszanie działania wyszukiwarek internetowych**
- **Zastosowanie w sieciach komputerowych**
  FPGA są wykorzystywane w przełącznikach sieciowych (switchach Ethernet) do realizacji szybkiego i konfigurowalnego przełączania pakietów.
- **Zastosowania naukowe**
  Układy FPGA są używane w laboratoriach badawczych, takich jak CERN, do analizy danych z detektorów cząstek. Przykładowo, służą one do wykrywania i rejestrowania zderzeń hadronów, przetwarzając dane z bardzo dużą szybkością w czasie rzeczywistym.

Konkretne zastosowanie omawianego zadania: termostat

Zasada działania:

- Na lewym wyświetlaczu ustawia się żądaną temperaturę (np. 3 = 30°C), wciskając lewy przycisk.
- Gdy wybierzesz odpowiednią wartość, wciskasz prawy przycisk.
- Wybrana wartość zostaje zatwierdzona i pokazana na prawym wyświetlaczu – to oznacza, że piec ma teraz utrzymać tę temperaturę.



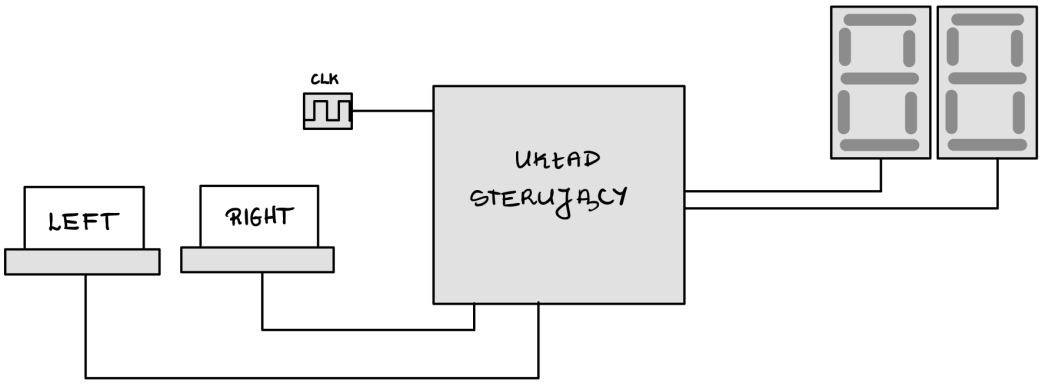
Rys. 1. Przykładowe zastosowanie omawianego układu

Idea rozwiązania

Kod został napisany przy użyciu języka VHDL. Interfejs menu realizowany jest przez moduł menu_selector napisany w architekturze Behavioral. Każde naciśnięcie lewego przycisku zwiększa cyfrę na lewym wyświetlaczu o 1 (modulo 10). Naciśnięcie prawego przycisku kopiuje cyfrę z lewego wyświetlacza do prawego. Działanie uwzględnia eliminację drgań styków, dzięki czemu przyciski nie generują fałszywych impulsów.

Moduł menu_selector posiada następujące porty:

- clk – zegar systemowy (typu std_logic), wykorzystywany do synchronizacji działań.
- left_button – wejście przycisku odpowiedzialnego za inkrementację wartości lewego wyświetlacza.
- right_button – wejście przycisku odpowiedzialnego za przypisanie wartości z lewego wyświetlacza do prawego.
- display_left – 7-bitowe wyjście odpowiadające za sterowanie lewym wyświetlaczem 7-segmentowym.
- display_right – 7-bitowe wyjście odpowiadające za sterowanie prawym wyświetlaczem 7-segmentowym.



Rys. 2. Schemat projektowanego układu

Wewnątrz architektury zdefiniowano stałe i sygnały:

- left_digit, right_digit – liczby całkowite (0–9) reprezentujące aktualnie wyświetlane cyfry.
- DEBOUNCE_COUNT – wartość określająca liczbę cykli zegara potrzebnych do potwierdzenia zmiany stanu przycisku (10000)
- debounce_counter0, debounce_counter1 – liczniki eliminujące drgania styków odpowiednio dla lewego i prawego przycisku.
- stable_left_button, stable_right_button – stabilny (odfiltrowany) stan przycisku.
- prev_stable_left_button, prev_stable_right_button – poprzedni stan przycisku, wykorzystywany do detekcji zbocza opadającego (naciśnięcia przycisku).

Procesy które działają w ranach menu_selector to ButtonProc, SegLeft i SegRight.

ButtonProc- proces obsługi przycisków i debouncowania. Działa przy każdym zboczu narastającym sygnału zegarowego clk i odpowiada za:

- Eliminację drgań styków przycisków:
  - Gdy stan przycisku się nie zmienia, licznik resetowany jest do zera.
  - Gdy wykryta zostanie zmiana, licznik inkrementuje się.
  - Po osiągnięciu limitu DEBOUNCE_COUNT, nowy stan uznawany jest za stabilny.
- Detekcję naciśnięcia przycisku (zbocze opadające: '1' → '0'):
  - left_button:
    - Inkrementuje left_digit.
    - Po osiągnięciu wartości 9, licznik zawija się do 0 (działanie cykliczne).
  - right_button:
    - Przypisuje aktualną wartość z left_digit do right_digit.
- Aktualizacja poprzednich stanów przycisków (konieczne do detekcji zboczy).

SegLeft i SegRight- procesy sterujące wyświetlaczami. Wykonują konwersję wartości liczbowych (0–9) na odpowiadające im kody segmentów wyświetlacza 7-segmentowego w układzie common cathode (logika aktywna niskim stanem logicznym '0'):

- 0 odpowiada kodowi "0000001"
- 1 odpowiada kodowi "1001111"
- ...
- 9 odpowiada kodowi "0000100"

Dla wartości spoza zakresu (chociaż nie powinny się pojawić), wyświetlany jest kod "1111111", który gasi wszystkie segmenty.

Kod źródłowy menu_selector.vhd:

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

entity menu_selector is

    ----------------------------------------------------------------------------
    -- Interfejs
    ----------------------------------------------------------------------------
    Port (
        clk           : in  std_logic; -- wejściowy sygnał zegarowy
        left_button   : in  std_logic; -- lewy przycisk
        right_button  : in  std_logic; -- prawy przycisk
        display_left  : out std_logic_vector(6 downto 0);  -- wyjście dla lewego
wyświetlacza
        display_right : out std_logic_vector(6 downto 0) -- wyjście dla prawego
wyświetlacza
    );
end menu_selector;



----------------------------------------------------------------------------
-- Logika działania
----------------------------------------------------------------------------
architecture Behavioral of menu_selector is

    ----------------------------------------------------------------------------
    -- Sygnały wewnętrzne
    ----------------------------------------------------------------------------
    signal left_digit   : integer range 0 to 9 := 0; -- liczba wyświetlana na lewym
wyświetlaczu
    signal right_digit  : integer range 0 to 9 := 0; -- liczba wyświetlana na prawym
wyświetlaczu

    -- stałe do debouncingu
    constant DEBOUNCE_COUNT : integer := 10000; -- minimalna liczba cykli zegara do
uznania zmiany stanu
    constant DEBOUNCE_LIMIT : integer := DEBOUNCE_COUNT + 1; -- maksymalna wartość
licznika

    signal debounce_counter0 : integer range 0 to DEBOUNCE_LIMIT := 0; -- licznik
eliminacji drgań dla lewego przycisku
```

```
   signal debounce_counter1 : integer range 0 to DEBOUNCE_LIMIT := 0; -- licznik
eliminacji drgań dla prawego przycisku

   signal stable_left_button      : STD_LOGIC := '1'; -- stabilny stan lewego
przycisku
   signal stable_right_button     : STD_LOGIC := '1'; -- stabilny stan prawego
przycisku
   signal prev_stable_left_button : STD_LOGIC := '1'; -- poprzedni stabilny stan
lewego przycisku
   signal prev_stable_right_button: STD_LOGIC := '1'; -- poprzedni stabilny stan
prawego przycisku
begin

   ---------------------------------------------------------------------------
   -- Proces obsługi przycisków z debounce i logiką wyboru
   ---------------------------------------------------------------------------
   ButtonProc: process(clk) -- proces obsługi przycisków, wyzwalany na zboczu
rosnącym zegara
   begin
       if rising_edge(clk) then
           -- debounce left_button
           if left_button = stable_left_button then -- jeśli stan lewego przycisku
jest taki, jak poprzedni stabliny
               debounce_counter0 <= 0; -- resetuj licznik
           else -- aktualny stan lewego przycisku różni się od poprzedniego
stabilnego
               debounce_counter0 <= debounce_counter0 + 1; -- zwiększ licznik
               if debounce_counter0 = DEBOUNCE_COUNT then -- jeśli licznik osiągnie
wartość graniczną (nowy stan trwa wystarczająco długo)
                   stable_left_button <= left_button; -- zmiana stanu stabilnego
lewego przycisku
                   debounce_counter0 <= 0; -- resetuj licznik
               end if;
           end if;

           -- debounce right_button- analogicznie do lewego przycisku
           if right_button = stable_right_button then
               debounce_counter1 <= 0;
           else
               debounce_counter1 <= debounce_counter1 + 1;
               if debounce_counter1 = DEBOUNCE_COUNT then
                   stable_right_button <= right_button;
                   debounce_counter1 <= 0;
               end if;
           end if;
```

```
            -- detekcja zbocza opadającego (active-low: '1' > '0')- sprawdzamy, czy
nastąpiło zbocze opadające
            if prev_stable_left_button = '1' and stable_left_button = '0' then --
jeśli nastąpiło zbocze opadajace dla lewego przycisku
                -- zmiana wartości lewego wyświetlacza
                if left_digit = 9 then
                    left_digit <= 0;
                else
                    left_digit <= left_digit + 1;
                end if;
            end if;

            if prev_stable_right_button = '1' and stable_right_button = '0' then --
jeśli nastąpiło zbocze opadajace dla prawego przycisku
                right_digit <= left_digit; -- przepisujemy wartość lewego
wyświetlacza na prawy
            end if;

            -- aktualizacja poprzednich stabilnych stanów przycisków
            prev_stable_left_button <= stable_left_button;
            prev_stable_right_button <= stable_right_button;
        end if;
    end process ButtonProc;

    ---------------------------------------------------------------------------
    -- Proces mapujący left_digit -> display_left (7-segment, active low- zapalenie
segmentu przy stanie '0')
    ---------------------------------------------------------------------------
    SegLeft: process(left_digit) -- proces wyzwalany zmianą left_digit
    begin
        case left_digit is
            when 0 => display_left <= "0000001";
            when 1 => display_left <= "1001111";
            when 2 => display_left <= "0010010";
            when 3 => display_left <= "0000110";
            when 4 => display_left <= "1001100";
            when 5 => display_left <= "0100100";
            when 6 => display_left <= "0100000";
            when 7 => display_left <= "0001111";
            when 8 => display_left <= "0000000";
            when 9 => display_left <= "0000100";
            when others => display_left <= "1111111";
        end case;
    end process SegLeft;
```

```
   ----------------------------------------------------------------------------
   -- Proces mapujący right_digit -> display_right (7-segment, active low-
zapalenie segmentu przy stanie '0')
   ----------------------------------------------------------------------------
   SegRight: process(right_digit) -- proces wyzwalany zmianą right_digit
   begin
       case right_digit is
           when 0 => display_right <= "0000001";
           when 1 => display_right <= "1001111";
           when 2 => display_right <= "0010010";
           when 3 => display_right <= "0000110";
           when 4 => display_right <= "1001100";
           when 5 => display_right <= "0100100";
           when 6 => display_right <= "0100000";
           when 7 => display_right <= "0001111";
           when 8 => display_right <= "0000000";
           when 9 => display_right <= "0000100";
           when others => display_right <= "1111111";
       end case;
   end process SegRight;

end Behavioral;
```

Mapowanie wejść i wyjść układu na piny płytki:

	Node Name	Direction	Location
1	clk	Input	PIN_91
2	display_left[6]	Output	PIN_6
3	display_left[5]	Output	PIN_7
4	display_left[4]	Output	PIN_8
5	display_left[3]	Output	PIN_9
6	display_left[2]	Output	PIN_11
7	display_left[1]	Output	PIN_12
8	display_left[0]	Output	PIN_13
9	display_right[6]	Output	PIN_17
10	display_right[5]	Output	PIN_18
11	display_right[4]	Output	PIN_19
12	display_right[3]	Output	PIN_20
13	display_right[2]	Output	PIN_21
14	display_right[1]	Output	PIN_23
15	display_right[0]	Output	PIN_24
16	left_button	Input	PIN_28
17	right_button	Input	PIN_29
18	<<new node>>		