Wydział Fizyki, Astronomii i Informatyki Stosowanej
Uniwersytetu Mikołaja Kopernika

Pracownia układów programowalnych

Ćwiczenie 4

Projekt i implementacja procesora PicoBlaze w strukturze Spartan-3E
(Spartan-3E Starter Kit, PicoBlaze, KCPSM, pBlazIDE)

Cel ćwiczenia

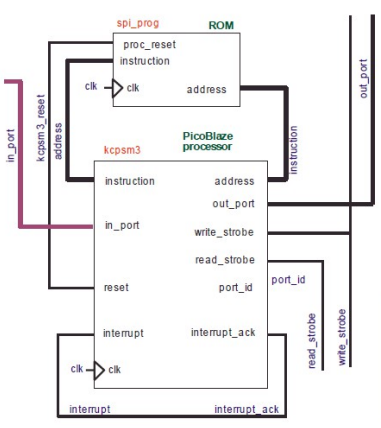
Nabycie umiejętności w posługiwaniu się oprogramowaniem ISE9.1. Zapoznanie się z zestawem prototypowym Spartan-3E FPGA Starter Kit, a następnie wykonanie projektu układu modulatora PWM ze zmiennym wypełnieniem okresu, stosując do tego celu procesor PicoBlaze [1,2].

Zagadnienia do przygotowania

Przed przystąpieniem do ćwiczenia student powinien posiadać przygotowany **wcześniej** algorytm eliminujący wpływ drgań styków mikroprzełączników oraz propozycję implementacji tego algorytmu do struktury programowalnej. Wykonujący ćwiczenie powinien posiadać wiedzę na temat budowy i własności układów FPGA, a w szczególności rodziny Spartan-3E [3]. Wymagana jest także podstawowa wiedza odnośnie modulacji **PWM** oraz budowy i zasady działania procesora PicoBlaze [6,7]. Ponadto należy zaznajomić się z dokumentacją dotyczącą zestawu laboratoryjnego Spartan-3E FPGA Starter Kit oraz zaczerpnąć informacji odnośnie podstawowych instrukcji procesora i obsługi środowiska graficznego pBlazIDE [4,5].

Przebieg ćwiczenia

1. Wykonać prosty test zestawu laboratoryjnego. Uruchomić w tym celu narzędzie Impact i po wcześniejszej inicjalizacji interfejsu Jtag zaprogramować układ XC3S500E dostarczonym do ćwiczenia plikiem s3esk_startup.bit [8]. Zaobserwować na zestawie stan diod w zależności od aktualnego stanu przełączników i przycisku obrotowego (enkodera).
2. Uruchomić zintegrowane środowisko projektowe ISE9.1 (Project Navigator), utworzyć nowy projekt (wybrać odpowiedni typ układu i projektu: Spartan-3E, Vhdl) oraz dodać nowe źródło do projektu (moduł vhdl), w którym zdefiniować dwa wejścia i jedno wyjście (wejścia: clk, rst; wyjście: LED). Dodać do projektu dwa pliki: kcpsm3.vhd oraz diodapBl.vhd [10]. Bazując na poniższym rysunku dokonać właściwego mapowania obu komponentów. W tym celu wymagane jest zadeklarowanie odpowiednich sygnałów.



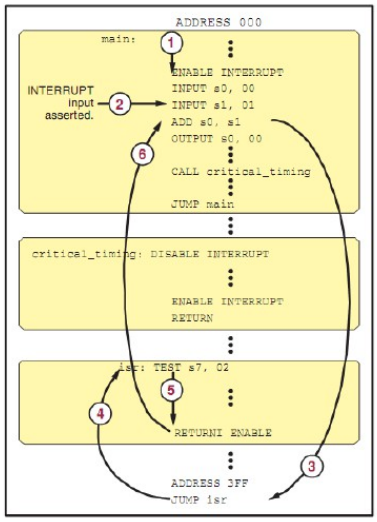

```
component KCPSM3
port (
      address      : out std_logic_vector( 9 downto 0);
      instruction  : in  std_logic_vector(17 downto 0);
      port_id      : out std_logic_vector( 7 downto 0);
      write_strobe : out std_logic;
      out_port     : out std_logic_vector( 7 downto 0);
      read_strobe  : out std_logic;
      in_port      : in  std_logic_vector( 7 downto 0);
      interrupt    : in  std_logic;
      interrupt_ack: out std_logic;
      reset        : in  std_logic;
      clk          : in  std_logic
     );
end component;
```

```
component prog_rom
port (
      address     : in  std_logic_vector( 9 downto 0);
      instruction : out std_logic_vector(17 downto 0);
      clk         : in  std_logic
     );
end component;
```

4. Sprawdzić w dokumentacji wymaganą częstotliwość pracy procesora PicoBlaze. W razie konieczności należy zaprojektować odpowiedni dzielnik częstotliwości.
5. Ponieważ liczba dostępnych portów we/wy zależy od portu port_id, koniecznym jest wybór właściwego z portów do którego w danej chwili mamy zamiar przesyłać informację. Dlatego też w pliku nadrzędnym projektu należy dokonać odpowiedniej implementacji takiego modułu. Przykładowy kod (w języku vhdl) realizujący powyższą czynność może przyjąć następującą postać:

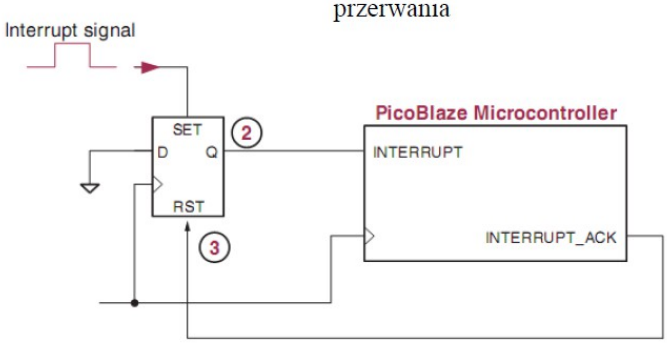
```
LED_ustaw: process(clk_p, rst)
begin
        if (rst = '1') then
                LED <= '0';
        elsif clk_p'event and clk_p = '1' then
                if(( port_id = "00000001") and (write_strobe = '1')) then
                        LED <= out_port(0);
                end if;
        end if;
end process;
```

6. Posiłkując się dokumentacją dokonać analizy powyższego zapisu i porównać z implementacją sprzętową zwracając szczególną uwagę na procedurę zapisu do portu mikroprocesora PicoBlaze.
7. Przeprowadzić symulację logiczną projektu przy użyciu dostępnych narzędzi dostarczonych przez producenta oprogramowania (ISESimulator).
8. Korzystając z załączonej do ćwiczenia dokumentacji (digilent_user_guide.pdf), utworzyć i dodać do projektu plik ograniczeń *.ucf. Następnie dokonać implementacji projektu w strukturę Spartan-3E oraz sprawdzić działanie.
9. W odrębnym katalogu uruchomić środowisko graficzne pBlazeIDE. Bazując na dołączonym pliku źródłowym diodapBlazIDE.psm (który należy dogłębnie przeanalizować), utworzyć nowe źródło i dokonać jego modyfikacji w taki sposób, aby w środowisku graficznym programu pBlazeIDE ukazały się porty we/wy mikroprocesora PicoBlaze [10]. Następnie w środowisku pBlazeIDE dokonać procesu symulacji projektu obserwując stan wszystkich zasobów procesora PicoBlaze (flagi, porty, rejestry). Prawidłowo wykonany projekt umożliwi wytworzenie pliku vhdl z odpowiednio zainicjalizowaną pamięcią ROM (właściwy kod programu), który należy następnie dodać do projektu. Dokonać implementacji projektu i sprawdzić działanie.
10. Samodzielnie zmodyfikować projekt w taki sposób, aby realizował on projekt wędrujących trzech diod LED (tzw. efekt węża świetlnego). Zwiększyć funkcjonalność projektu wprowadzając dodatkowo odczyt portu wejściowego mikrokontrolera. Na podstawie odczytanego stanu wybranego z dostępnych w zestawie przełączników, dokonać wyboru kierunku świecenia poszczególnych diod. Niezbędne informacje odnośnie wymaganego opisu sprzętowego projektu zamieszczono w załączonych do ćwiczenia dokumetacjach.
11. Zmodyfikować istniejący projekt wprowadzając dodatkowo możliwość obsługi przerwania. Niech przerwanie nastąpi w wyniku wciśnięcia przycisku sw (wymagana jest w tym miejscu implementacja układu likwidującego drgania styków mikroprzełączników [9]). W funkcji obsługi przerwania dokonać odczytu stanu portu wejściowego mikroprocesora PicoBlaze. Podobnie jak wcześniej niezbędne informacje odnośnie wymaganego opisu sprzętowego projektu zamieszczono w dołączonych do ćwiczenia dokumetacjach.

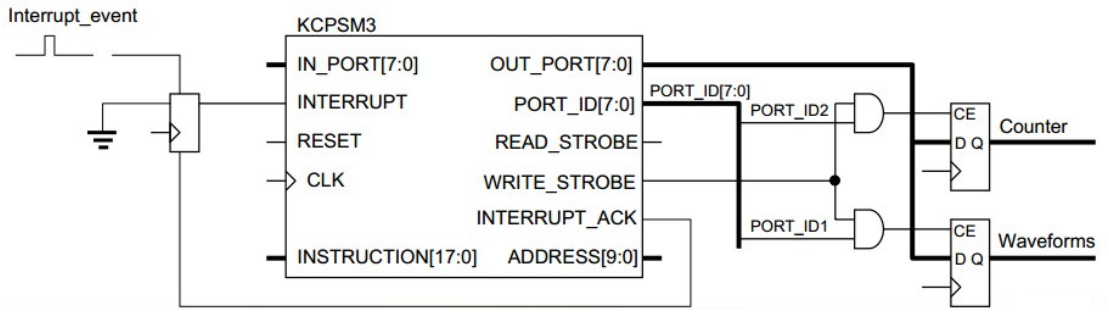


Podstawowe własności:

ENABLE INTERRUPT - Flaga przerwań
$3FF adress instrukcji przerwania
RETURNI – wyjście z procedury obsługi przerwania



12. *(punkt obowiązkowy dla 60-cio godzinnych grup ćwiczeniowych)* Zrealizować samodzielnie projekt modulatora PWM ze zmiennym wypełnieniem okresu. Wypełnienie określane jest na podstawie aktualnego stanu przełącznika bądź przełączników (odczytywanych przez port IN_PORT w funkcji obsługi przerwania wywołanej dowolnie wybranym przyciskiem sw). Początkową częstotliwość wytwarzanego przebiegu ustalić na 1Hz. Zaimplementować projekt w strukturę programowalną i przetestować jego działanie. Schemat poglądowy, który może posłużyć jako szkic projektu przedstawiono na poniższym rysunku:



Licznik Counter pokazuje procentową wartość wypełnienia, natomiast wyjście Waveforms jest wyjściem modulatora PWM.

Literatura

[1] Język VHDL, Projektowanie programowalnych układów logicznych. Kevin Skahill. WNT, Warszawa.
[2] Marcin Nowakowski, PicoBlaze. Mikroprocesor w FPGA, BTC, 2009.
[3] Spartan-3E FPGA Family Data Sheet (ds312.pdf).
[4] Spartan-3E Starter Kit Board User Guide (digilent_user_guide.pdf).
[5] Schemat płytki prototypowej z opisem wyprowadzeń (digilent_sch.pdf).
[6] PicoBlaze 8-bit Embedded Microcontroller User Guide (ug129.pdf),
[7] KCPSM3 8-bit Micro Controller for Spartan-3, Virtex-II and Virtex-IIPRO, 2003 (KCPSM_Manual.pdf),
[8] PicoBlaze, Initial Design for Spartan-3E Starter Kit, 2006 (s3esk_startup.pdf),
[9] Eliminacja drgań styków mikroprzełaczników (debouncer.pdf)
[10] Pliki źródłowe: kcpsm3.vhd, diodapBl.vhd, diodapBlazIDE.psm

UWAGA, pozycje literaturowe [3-8] dostępne są w wersji elektronicznej (pliki PDF).

instrukcję do ćwiczenia przygotował Robert Frankowski