Systemy Czasu Rzeczywistego

„FPGA”

laboratorium: 05

autor: mgr inż. Mateusz Baran

Data utworzenia: 01/10/14

1 Spis treści

2 Wiadomości wstępne

Pierwsza część niniejszej instrukcji zawiera podstawowe wiadomości teoretyczne dotyczące omawianego tematu. Poznanie tych wiadomości umożliwi prawidłowe zrealizowanie praktycznej części laboratorium.

2.1 Niezbędne wiadomości

Przed laboratorium należy zapoznać się z podstawowymi pojęciami dotyczącymi budowy układów FPGA:

- zasoby logiczne FPGA (CLB, slice, pamięć Block RAM, mnożarki),
- zasoby połączeniowe (programowalne połączenia wewnętrze, IOB),
- podstawy języka VHDL.

Podczas laboratorium będzie wykorzystywany układ Digilent Basys2 oparty o układ Xilinx Spartan3E-250.

2.2 Literatura:

[1] Podstawowy opis architektury układów FPGA:
http://www.csd.uoc.gr/~hy220/2009f/lectures/11_basic_fpga_arch.pdf

[2] Szczegóły budowy wykorzystywanych układów FPGA:
http://www.xilinx.com/support/documentation/data_sheets/ds312.pdf

[3] Podstawy języka VHDL:
https://wiki.ittc.ku.edu/ittc/images/3/37/EECS_140_VHDL_Tutorial.pdf

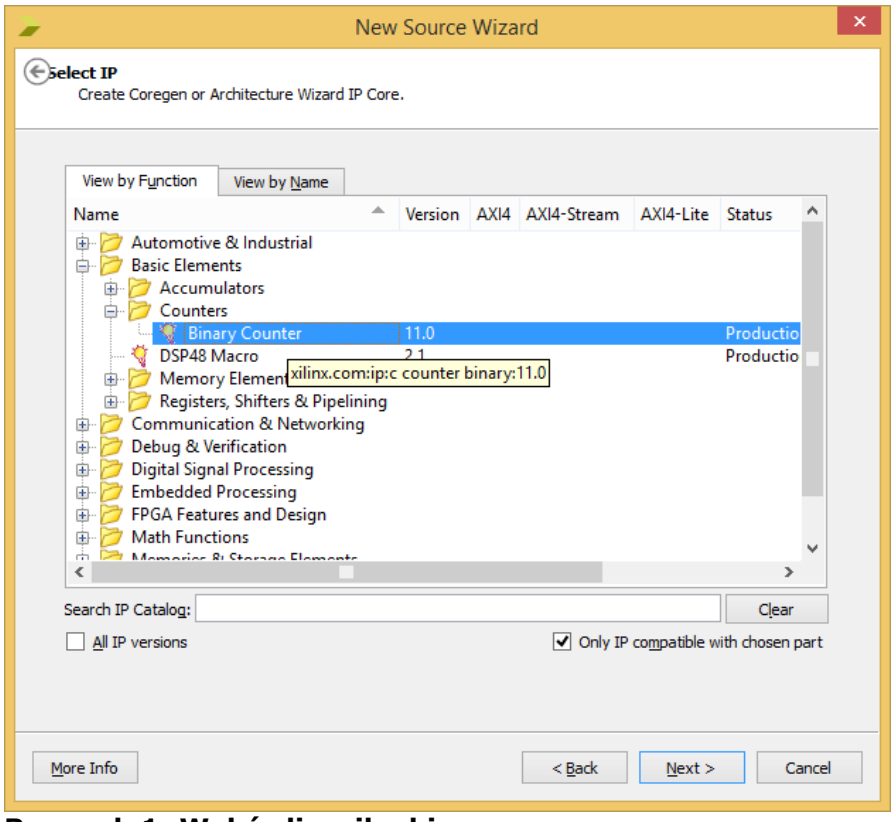
3 Przebieg laboratorium

3.1 Zadanie 1. Na ocenę 3.0 (dst)

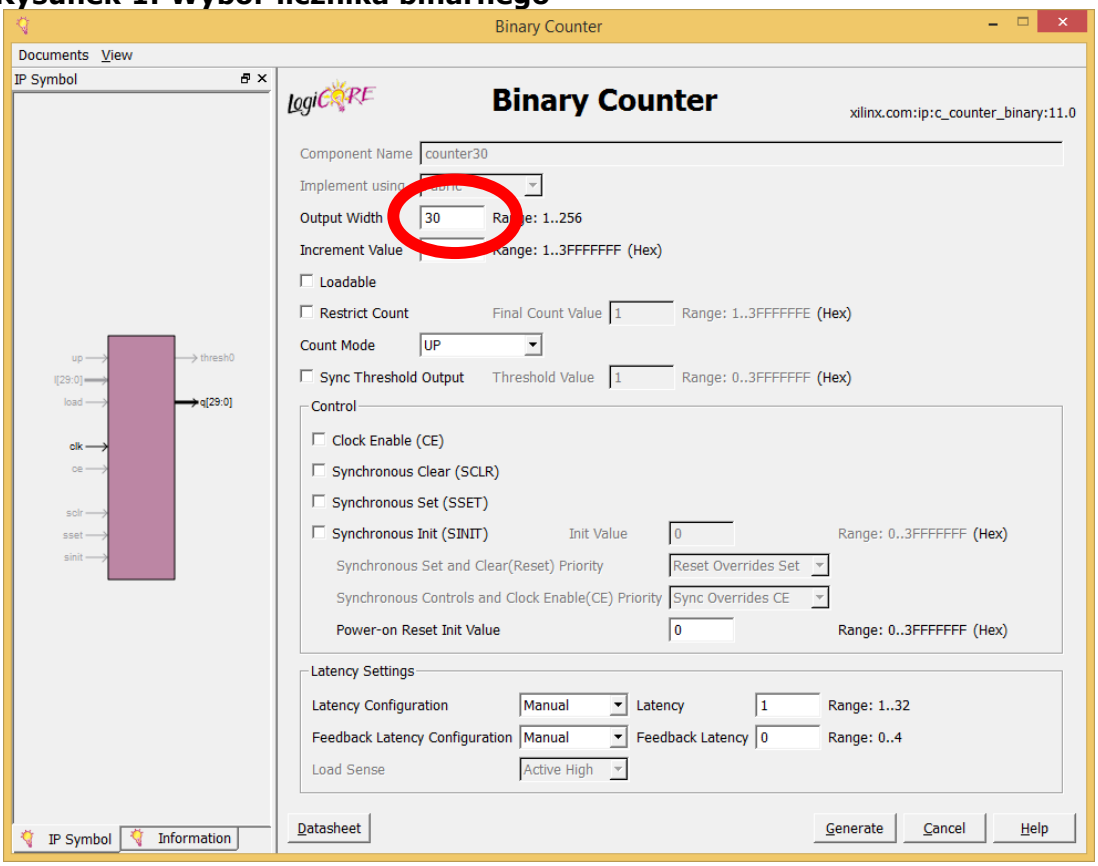
To zadanie wprowadza kilka nowych elementów. Pierwszym z nich jest użycie pamięci BlockRAM w projekcie układu. Proszę utworzyć nowy projekt o nazwie FlashyLights, podobnie jak w pierwszym laboratorium. Proszę dodać do projektu plik źródłowy o nazwie FlashyLights z wejściem zegarowym oraz magistralą wyjściową na diody LED. Należy pamiętać o utworzeniu pliku ograniczeń z odpowiednimi ustawieniami.

W kolejnym kroku proszę dodać nowe źródło typu IP (Core Generator & Architecture Wizard) o nazwie counter30. Rdzenie IP są gotowymi modułami które można wykorzystywać do budowy własnych projektów. Przy wyborze rodzaju rdzenia proszę zaznaczyć „Only IP compatible with chosen part” i odnaleźć licznik binarny (Binary Counter) – patrz Rysunek 1. Następnie należy kliknąć „Dalej” i „Zakończ”. Po chwili pokaże się okno konfiguracji. Należy przestawić tam szerokość wyjścia (Output Width) na 30.

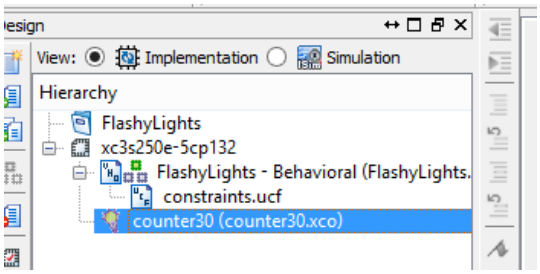
Generowanie licznika, wykonywane po ustawieniu parametrów, może zająć nawet kilka minut. Po tym czasie odpowiedni wpis powinien pojawić się w hierarchii projekt (patrz Rysunek 3). Wykorzystanie takiego licznika odbywa się na takiej samej zasadzie jak dowolnego innego modułu. Jest to prezentowane w poniższym kodzie.



Rysunek 1: Wybór licznika binarnego



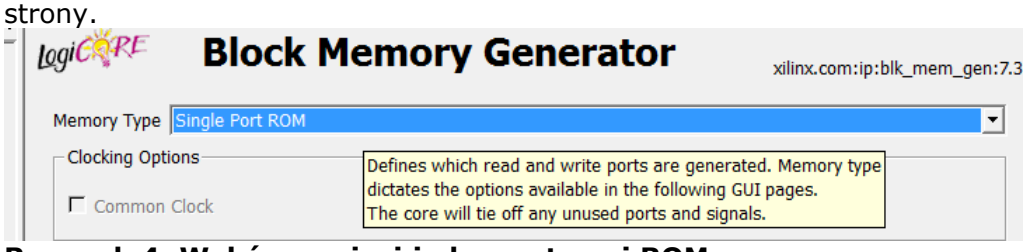
Rysunek 2: Ustawianie szerokości wyjścia licznika



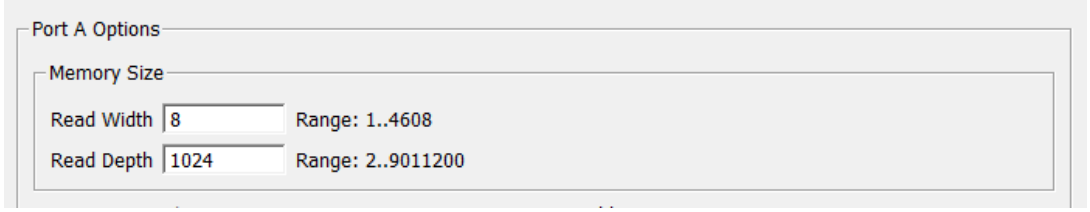
Rysunek 3: Licznik w hierarchii projektu

```
architecture Behavioral of FlashyLights is
 COMPONENT counter30
  PORT (
   clk : IN STD_LOGIC;
   q : OUT STD_LOGIC_VECTOR(29 DOWNTO 0)
  );
 END COMPONENT;

 signal count : STD_LOGIC_VECTOR(29 downto 0);
begin

addr_counter : counter30
 PORT MAP (
  clk => clk,
  q => count
 );

end Behavioral;
```

Teraz do projektu zostanie dołożona pamięć. Proszę dodać kolejny rdzeń IP **o nazwie memory** typu „Block memory generator”. W ustawieniach należy wybrać jednoportową pamięć ROM (druga strona, Rysunek 4) oraz ustawić szerokość i głębokość jak na Rysunek 5. Na czwartej stronie należy wybrać plik inicjalizujący pamięć (Flashy.coe), jak na Rysunek 6. Następnie należy wcisnąć „generate” omijając kolejne strony.



Rysunek 4: Wybór pamięci jednoportowej ROM



Rysunek 5: Szerokość i głębokość pamięci

Rysunek 6: Plik inicjalizujący pamięć

Teraz należy podłączyć pamięć. Powinna być ona zadeklarowana następująco:

```
COMPONENT memory
PORT (
  clka : IN STD_LOGIC;
  addra : IN STD_LOGIC_VECTOR(9 DOWNTO 0);
  douta : OUT STD_LOGIC_VECTOR(7 DOWNTO 0)
);
END COMPONENT;
```

Zegar clka należy podłączyć do sygnału zegarowego, wyjście danych (douta) do diod LED, zaś wejście adresowe (addra) do dziesięciu najstarszych bitów licznika binarnego. Proszę zamieścić schemat RTL w sprawozdaniu.

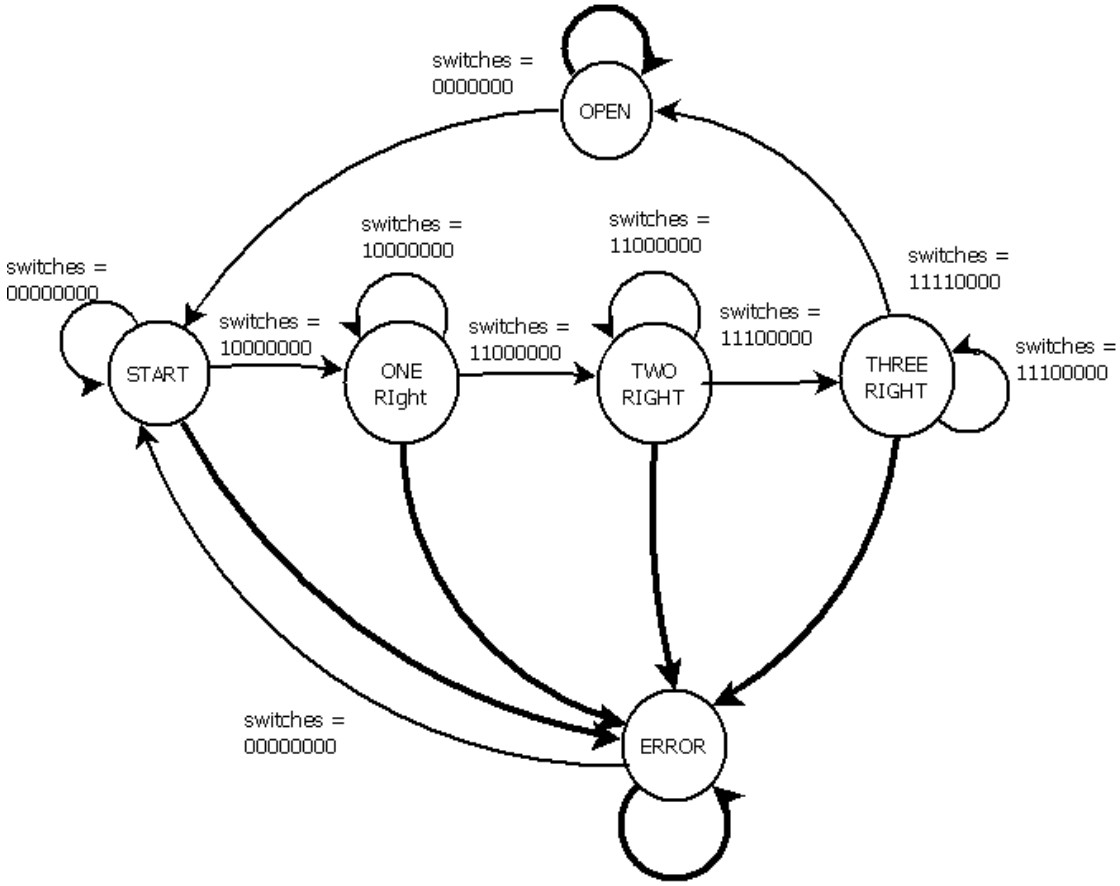
Po wykonaniu instrukcji proszę zobaczyć efekt na płytce z układem FPGA. Obejrzyj opcje konfiguracji licznika. Co należy zmienić, aby licznik liczył w dół? Jak ograniczyć zakres wartości po których przebiega licznik? Zapisz nazwy odpowiednich opcji w raporcie.

3.2 Zadanie 2. Na ocenę 4.0 (db)

Zadanie drugie polega na napisaniu projektu realizującego maszynę stanów skończonych. Maszyna ta będzie przechowywała swój stan za pomocą magistrali i zmieniała go gdy użytkownik zmieni stan przełącznika. Projekt można oprzeć o wcześniejsze zadanie Switches_LEDs.

Projektowaną maszynę przedstawia Rysunek 7. Ma ona sześć stanów. Maszyna zaczyna działanie od stanu START. Włączanie kolejnych przełączników przenosi ją w kolejne stany na prawo od stanu START (ONE RIGHT, TWO RIGHT, THREE RIGHT). Gdy maszyna jest w stanie THREE RIGHT i użytkownik przełączy czwarty przełącznik, maszyna przechodzi do stanu OPEN. Ze stanu OPEN można przejść ponownie do stanu start poprzez umieszczenie wszystkich przełączników w pozycji ‘0’. Wszystkie nieopisane tu zmiany przełączników skutkują przejściem do stanu ERROR, z którego można przejść jedynie do stanu START poprzez umieszczenie wszystkich przełączników w pozycji ‘0’.

Działanie maszyny ma być obrazowane stanem diod LED. Te sterowane sygnałami 7 do 4 powinny obrazować numer stanu maszyny (podane są one poniżej), zaś diody 3 do 0 powinny być zapalone w stanie OPEN i zgaszone we wszystkich pozostałych.



Rysunek 7: Maszyna stanów skończonych

Stan maszyny będzie reprezentowany przez sygnał „state”, zaś wartości odpowiadające poszczególnym stanom będą zdefiniowane przez odpowiednie stałe:

```
constant state_error     : STD_LOGIC_VECTOR(3 downto 0) := "0000";
constant state_start     : STD_LOGIC_VECTOR(3 downto 0) := "0001";
constant state_one_right : STD_LOGIC_VECTOR(3 downto 0) := "0010";
constant state_two_right : STD_LOGIC_VECTOR(3 downto 0) := "0011";
constant state_three_right : STD_LOGIC_VECTOR(3 downto 0) := "0100";
constant state_open : STD_LOGIC_VECTOR(3 downto 0) := "0101";

signal state : STD_LOGIC_VECTOR(3 downto 0) := state_start;
```

Przejścia maszyny stanów skończonych będziemy realizować za pomocą odpowiedniego procesu:

```
process (clk)
begin
   if rising_edge(clk) then
   case state is
     when state_error =>
       leds(7 downto 4) <= "0000";
       case switches is
       when "00000000" => state <= state_start;
```



```
        when others     => state <= state_error;
        end case;
      when state_start =>
…
      when state_one_right =>
…
      when state_two_right =>
…
      when state_three_right =>
…
      when state_open =>
…
      when others =>
        leds(7 downto 4) <= "0000";
        state <= state_error;
    end case;
 end if;


          end process;
```

Proszę uzupełnić powyższy kod tak, aby odpowiadał opisowi maszyny stanów skończonych. Następnie należy uruchomić projekt w układzie FPGA i kilkukrotnie przetestować, czy działa. W zależności od szczegółów realizacji przełącznika, przez pewien krótki czas w trakcie przełączania zadawany sygnał może oscylować pomiędzy ‘0’ a ‘1’, zanim przyjmie nową wartość. Proszę zastanowić się w jaki sposób taka oscylacja może powodować złe działanie maszyny stanów.

3.3 Zadanie 3. Na ocenę 5.0 (bdb)

Zmodyfikuj maszynę stanów skończonych tak, aby opisane w poprzednim zadania wahania stanu przełącznika nie powodowały przejścia do stanu ERROR (w przypadku wykrycia wahania maszyna powinna pozostać w nowym stanie). W sprawozdaniu zamieść kod i schemat zmodyfikowanej maszyny stanów.

