Układy Cyfrowe I Systemy Wbudowane

Sprawozdanie z Laboratorium nr 6

Autorzy:

Kamil Kamyszek, 234932

Dagmara Szykulska, 235008

Prowadzący:

dr inż. Jacek Mazurkiewicz

05.12.2018

**1 Cel laboratorium**

Celem laboratorium było zapoznanie się z językiem opisu sprzętu jakim jest VHDL (Very High Speed Integrated Circuits Hardware Description Language). Nasza grupa otrzymała do stworzenia zamek szyfrowy, licznik synchroniczny mod 7 pozytywny w kodzie graya i translator kodu.

**2 Wykonane zadania**

2.1 Zamek Szyfrowy

Nasz zamek szyfrowy miał „otwierać się” po podaniu sekwencji 00011. Oto jak wyglądał kod napisany w VHDL-u:

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity zamek is

Port ( wej : in STD\_LOGIC; #wprowadzanie znaku

CLK : in STD\_LOGIC; #zegar

CE : in STD\_LOGIC; #jeśli wciśnięte to zamek nasłuchuje sekwencji”

RE : in STD\_LOGIC; #reset

wyj : out STD\_LOGIC); #wyjście

end zamek;

architecture Behavioral of zamek is

type state\_type is (A,B,C,D,E,F); #kolejne stany zamka

signal state, next\_state: state\_type;

begin

process1: process(CLK) #włączenie pierwszego procesu

begin

if rising\_edge(CLK) then #jeśli zegarek jest w stanie ‘1’

if RE='0' then #jeśli zresetowane to ->

if CE='1' then #jeśli wykrywanie sekwencji włączone to ->

state <= next\_state; #przechodzi w kolejny stan

end if;

else

state<=A;

end if;

end if;

end process process1;

process2: process(wej, state) #proces wykrywania sekwencji

begin

next\_state<=state;

case state is

when A =>

if wej='0' then #jeśli 0 to idzie do kolejnego stanu

next\_state <=B;

end if;

when B =>

if wej='0' then #jeśli 0 to idzie do kolejnego stanu

next\_state <=C; #itd…

else

next\_state <=A;

end if;

when C =>

if wej='0' then

next\_state <=D;

else

next\_state <=A;

end if;

when D =>

if wej='1' then

next\_state <=E;

else

next\_state <=C;

end if;

when E =>

if wej='1' then

next\_state <=F;

else

next\_state <=B;

end if;

when F =>

if wej='1' then

next\_state <=A;

else

next\_state <=B;

end if;

end case;

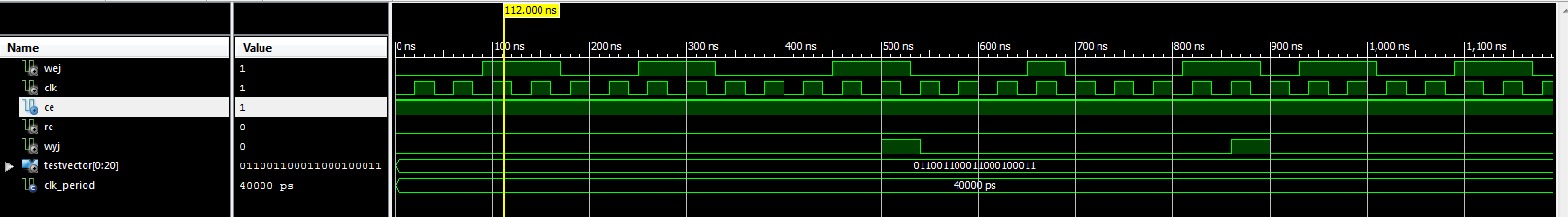
end process process2;

wyj <='1' when state=F else '0'; #jeśli sekwencja podana poprawnie otwiera zamek, w przeciwnym wypadku nie

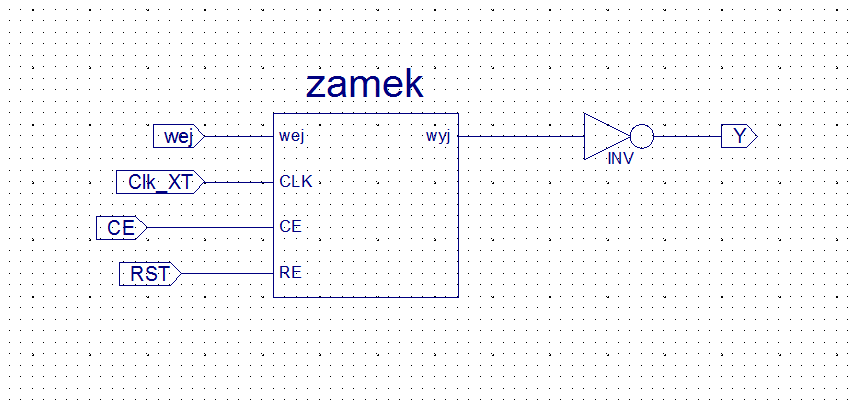
end Behavioral;

**Przebieg symulacji:**

Symulacja przebiegła poprawnie zgodnie z oczekiwaniami. W przypadku podania poprawnej sekwencji zamek się otwierał i zapalała się dioda led.



Mając działający kod VHDL mogliśmy wygenerować symbol, do którego wystarczyło podpiąć wejścia i wyjścia. Dzięki temu zabiegowi nasz schemat był niewielki i czytelny:



**2.2 Licznik synchroniczny mod 7 w kodzie graya**

W przypadku licznika mogliśmy zauważyć jak wiele mniej czasu jest potrzebne na stworzenie działającego układu pisząc w VHDL-u, w porównaniu z wykonaniem tego samego zadania używając tabel prawdy i siatek karnough.

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity sevenBitGrayCounter is

Port ( clr : in STD\_LOGIC; #reset

cnt : in STD\_LOGIC; # wykrywacz sekwencji

clk : in STD\_LOGIC; #zegar

q : out STD\_LOGIC\_VECTOR(2 downto 0) #bus dla 3 bitów (3 znaki)

); end sevenBitGrayCounter;

architecture gray\_cnt\_arch of sevenBitGrayCounter is

signal reg: STD\_LOGIC\_VECTOR(2 downto 0);

begin

process(clk) begin

if clk'event and clk = '1' then

if clr = '1' then reg <= "000";

elsif cnt = '1' then #jeśli licznik aktywowany

case reg is

when "000" => reg <= "001"; #kolejne stany licznika w kodzie graya

when "001" => reg <= "011"; #jeśli „liczba” to „kolejna liczba”

when "011" => reg <= "010";

when "010" => reg <= "110";

when "110" => reg <= "111";

when "111" => reg <= "101";

when "101" => reg <= "000";

when others => reg <= "000";

end case;

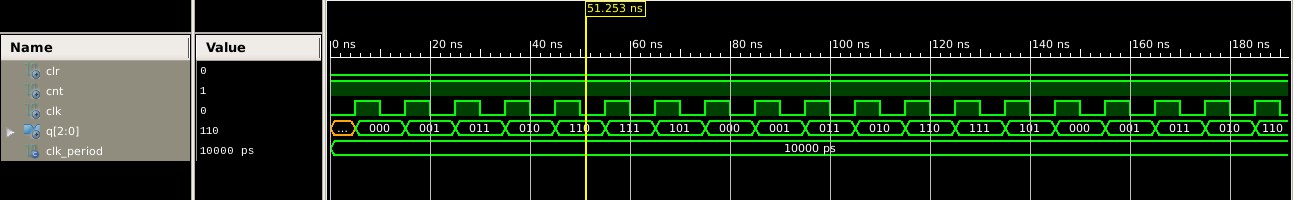
end if;

end process;

q <= reg;

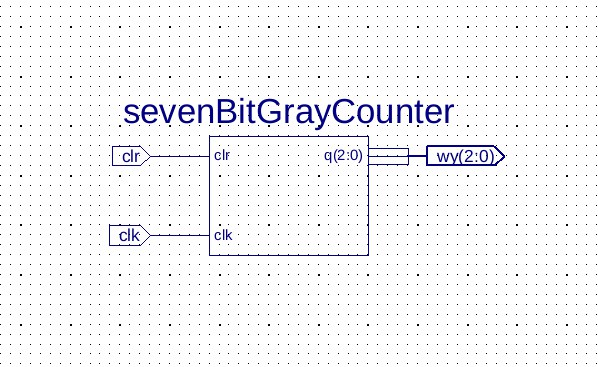
end gray\_cnt\_arch;

**Przebieg symulacji:**



**Schemat:**

Po rozmowie z prowadzącym postanowiliśmy pozbyć się wejścia „cnt” ze schematu, gdyż było ono nadmiarowe i nie wnosiło żadnej przydatnej funkcjonalności:



**2.3 Translator kodu binarnego na graya**

Niestety nie udało nam się wykonać tego zadania z powodu braku czasu. Zakończyliśmy to polecenie na poniżej napisanym kodzie, który jednak wyrzucał błędy:

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.STD\_LOGIC\_ARITH.ALL;

use IEEE.STD\_LOGIC\_UNSIGNED.ALL;

entity translatorKodu is

Port ( bin : in STD\_LOGIC\_VECTOR(3 downto 0);

gray : out STD\_LOGIC\_VECTOR (3 downto 0));

end translatorKodu;

architecture Behavioral of translatorKodu is

begin

bin(3) <= gray(3);

bin(2) <= gray(3) xor gray(2);

bin(1) <= gray(2) xor gray(1);

bin(0) <= gray(1) xor gray(0);

end Behavioral;

Log błędów:



**3 Wnioski**

Dwa zadania, które wykonaliśmy udało się załadować na przystawkę i przetestować ich działanie. Dzięki tym zajęciom nauczyliśmy się wykorzystywać język VHDL do tworzenia schematów różnych układów. Pozwali nam to tworzyć większe układy jednocześnie zmniejszając ilość czasu potrzebnego do ich wykonania.