Piotr Chmiel 200608 Układy Cyfrowe i Systemy Wbudowane 1

Piotr Sotor 200882 Data laboratorium: 17:05 18.12.2014

Prowadzący: mgr inż. Antoni Sterna

Sprawozdanie

Laboratorium nr 11

**Temat:** Płyta Spartan-3E Starter: zasoby podstawowe, praca z portami RS, PS/2

# Cele i założenia laboratorium:

Celem zajęć laboratoryjnych było zapoznanie się z nowym układem Spartan, zaprojektowanie prostych układów wykorzystujących port szeregowy oraz wyświetlacz lcd. Wykonanie układu wykorzystującego port PS/2 było niemożliwe z powodu braku wymaganej do tego myszki podłączanej poprzez PS/2. W ramach zajęć laboratoryjnych zaprojektowano i pomyślnie zaprogramowano następujące układy:

* Retransmisja na porcie szeregowym
* Retransmisja na porcie szeregowym z modyfikacją danych
* Konwerter hex -> dec z obsługą wyświetlacza LCD 1x64

# Synteza zadań

## Retransmisja na porcie szeregowym

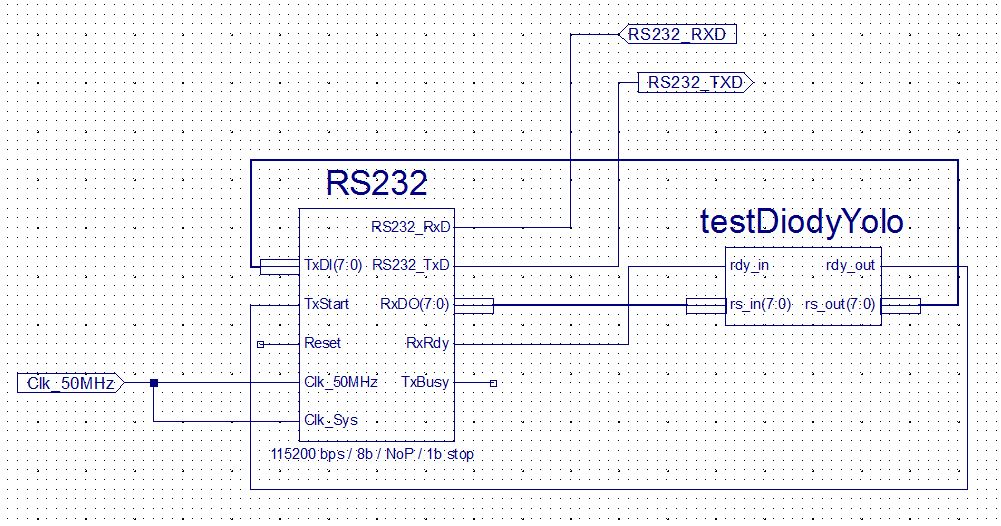
### Opis słowny

Celem tego zadania było zapoznanie się z obsługą modułu RS232 działającego zarówno jako odbiornik i nadajnik portu szeregowego. Najprostszą realizacją tego zadania byłoby proste podłączenie magistrali danych odebranych na magistralę wejściową danych wysyłanych, oraz wyjścia **RxRdy** na wejście **TxStart** – RxRdy sygnalizuje obiór bajtu, a TxStart rozpoczyna transmisję odebranego bajtu.

Znając treść drugiego zadania postanowiono utworzyć układ ułatwiający wejście w następne zadanie poprzez utworzenie pustego jeszcze interfejsu dla przyszłego modyfikatora danych retransmisji. Wspomniany pusty interfejs spełnia na razie zadanie opisane kilka linii wcześniej – bajt odebrany podaje na bajt wysyłany, sygnał odbioru bajtu podaje na sygnał startu transmisji. Pusty interfejs dla niepoznaki został nazwany testDiodyYolo – nazwa nie ma żadnego związku z jego funkcją.

Do prawidłowej obsług portu RS232 wymagane było dodanie do projektu pliku **GenIO.ucf.**

### Schemat układu



* + 1. **Implementacja interfejsu testDiodyYolo w VHDL**

|  |
| --- |
| library IEEE;  use IEEE.STD\_LOGIC\_1164.ALL;  use IEEE.NUMERIC\_STD.ALL;  use IEEE.std\_logic\_unsigned.ALL;  entity testDiodyYolo is  Port ( rs\_in : in STD\_LOGIC\_VECTOR(7 downto 0);  rdy\_in : in STD\_LOGIC;  rdy\_out : out STD\_LOGIC;  rs\_out : out STD\_LOGIC\_VECTOR(7 downto 0));  end testDiodyYolo;  architecture Behavioral of testDiodyYolo is  begin  rs\_out<=rs\_in;  rdy\_out<=rdy\_in;  end Behavioral; |

* + 1. **Zawartość pliku GenIO.ucf**

|  |
| --- |
| NET "Clk\_50MHz" LOC = "C9" | IOSTANDARD = LVTTL;  NET "Clk\_50MHz" PERIOD = 20.0ns HIGH 50%;  NET "RS232\_RXD" LOC = "R7" | IOSTANDARD = LVTTL ;  NET "RS232\_TXD" LOC = "M14" | IOSTANDARD = LVTTL | DRIVE = 8 | SLEW = SLOW ; |

## Retransmisja na porcie szeregowym z modyfikacją danych

### Opis słowny

Celem tego zadania było zaimplementowanie w VHDL wnętrza interfejsu **testDiodyYolo** tak by moduł ten w określony sposób modyfikował i filtrował bajty podlegające retransmisji .

Zaimplementowano modyfikację bajtów w formie zamiany kodów ASCII małych liter na wielkie oraz wielkich liter na małe. Filtracja została zaimplementowana jako pomijanie kodów ASCII reprezentujących cyfry.

W praktyce oznacza to pomijanie bajtów z zakresu 0x30 – 0x39, dodawanie stałej **c=0x20** do bajtów z zakresu **0x41(A) – 0x5a(Z)**, oraz odejmowanie tej samej stałej od bajtów z zakresu **0x61(a) – 0x7a(z)**, przy czym granice przedziałów należą do przedziałów.

Całość zrealizowano w formie układu kombinacyjnego, w którym pomijanie określonych bajtów zrealizowane jest poprze odpowiednie sterowanie wyjściem **rdy\_out**, a modyfikacje bajtu odpowiednimi operacjami arytmetycznymi

Schemat całego układu jak i zawartość pliku **GenIO.ucf** będzie niezmieniona względem pierwszego zadania więc nie zostaną przedstawione powtórnie.

* + 1. **Implementacja modułu testDiodyYolo w VHDL**

|  |
| --- |
| library IEEE;  use IEEE.STD\_LOGIC\_1164.ALL;  use IEEE.NUMERIC\_STD.ALL;  use IEEE.std\_logic\_unsigned.ALL;  entity testDiodyYolo is  Port ( rs\_in : in STD\_LOGIC\_VECTOR(7 downto 0);  rdy\_in : in STD\_LOGIC;  rdy\_out : out STD\_LOGIC;  rs\_out : out STD\_LOGIC\_VECTOR(7 downto 0));  end testDiodyYolo;  architecture Behavioral of testDiodyYolo is  begin  process (rdy\_in)  variable temp : STD\_LOGIC\_VECTOR (7 downto 0);  begin  temp := rs\_in;  if rs\_in < character'pos('0') or rs\_in > character'pos('9') then  rdy\_out <= rdy\_in;    if rs\_in >= character'pos('a') and rs\_in <= character'pos('z') then  temp := temp - 32;  end if;  if rs\_in >= character'pos('A') and rs\_in <= character'pos('Z') then  temp := temp + 32;  end if;  rs\_out <= temp;  else  rdy\_out <= '0';  end if;  end process;  end Behavioral; |

## Konwerter hex -> dec z obsługą wyświetlacza LCD 1x64

### Opis słowny

Początkowo celem tego ćwiczenia był odczyt danych z portu **PS/2**, do którego miała być podłączona mysz komputerowa, jednakże z braku takowej ograniczono zakres tego ćwiczenia do wyświetlenia na wyświetlaczu sfabrykowanych ręcznie wartości.

Założeniem ćwiczenia jest wyświetlanie liczb w postaci dziesiętnej, co niesie ze sobą problem konwersji **hex -> dec**. Do konwersji zaimplementowano algorytm **Double dabble**. Opis działania tego algorytmu wraz z przykładowym przebiegiem można łatwo odnaleźć na stronie angielskiej wersji portalu **Wikipedia.org** więc nie zostanie on tutaj opisany. Ważne z punktu widzenia ćwiczenia jest to, że 8-bitowa liczba hex konwertowana jest do 12-bitowej liczby BCD – kolejne czwórki bitów reprezentują kolejne cyfry zapisu dziesiętnego, np.:

11110011 (hex) -> 0010 0100 0011 (BCD)  
 (F3) (2) (4) (3)

Otrzymany w ten sposób zapis liczby w postaci **BCD** można bezpośrednio wykorzystać do obsługi wyświetlacza.

Zaimplementowany moduł **konw** reprezentujący wyżej opisany konwerter można w łatwy sposób przeskalować do obsługi wielobajtowych wartości przy zachowaniu dobrej czytelności kodu, co nie byłoby możliwe w przypadku implementacji konwertera jako dzielenia wartości przez kolejne potęgi liczby 10.

W celu zwiększenia czytelności wyświetlanych wartości na wyświetlaczy postanowiono wygasić pozycje wyświetlacza na które nie zostają podawane żadne wartości.

Do obsługi modułu wyświetlacza **LCD 1x64** wykorzystano gotowy moduł oraz zaimportowano odpowiednie pliki **\*.ucf** zawierający opis wejść i wyjść modułu – **LCD.ucf**, **GenIO.ucf**.

### Implementacja modułu konw - konwertera hex -> dec (BCD) – double dabble

|  |
| --- |
| library IEEE;  use IEEE.STD\_LOGIC\_1164.ALL;  use IEEE.NUMERIC\_STD.ALL;  use IEEE.std\_logic\_unsigned.ALL;  use IEEE.std\_logic\_arith.ALL;  entity konw is  Port ( wej : in STD\_LOGIC\_VECTOR(7 downto 0);  wyj : out STD\_LOGIC\_VECTOR(11 downto 0));  end konw;  architecture Behavioral of konw is  begin  process(wej)  variable temp : STD\_LOGIC\_VECTOR (11 downto 0 );  variable temp\_wej : STD\_LOGIC\_VECTOR (7 downto 0 );  variable licznik : INTEGER;  begin  licznik := 8;  temp := "000000000000";  temp\_wej := wej;    while licznik /= 0 loop    if temp (3 downto 0) > 4 then  temp := temp + 3;  end if;  if temp (7 downto 4) > 4 then  temp := temp + 48;  end if;  if temp (11 downto 8) > 4 then  temp := temp + 768;  end if;    temp(11 downto 1) := temp(10 downto 0);  temp(0) := temp\_wej(7);  temp\_wej(7 downto 1) := temp\_wej(6 downto 0);    licznik := licznik - 1;  end loop;    wyj <= temp;  end process;  end Behavioral; |

### Schemat układu

# Wnioski

Zajęcia miały mieć charakter zapoznawczy z nowym urządzeniem i taki też miały. Nie napotkano żadnych większych różnic pomiędzy programowaniem pod układ CPLD a układ FPGA. Jedynymi różnicami są potrzeba importowania więcej niż jednego pliku **\*.ucf** w zależności od potrzeb i obsługa programu iMPACT do wgrywania programu na urządzenie, co jest spowodowane większymi możliwościami samej płytki – więcej urządzeń wewnętrznych możliwych do niezależnego oprogramowania.