WOJSKOWA AKADEMIA TECHNICZNA
WYDZIAŁ ELEKTRONIKI

Laboratorium układów programowalnych	
Projekt w języku VHDL	
Temat projektu: **Uniwersalny przelicznik walut**	
Grupa: **E6T1N2**	Data wykonania ćwiczenia: **14.10.2006**
Wykonał: **Jacek SANIEWSKI**	Ocena:
	Prowadzący: **dr inż. Z. JACHNA**

1. WPROWADZENIE

Projekt uniwersalnego przelicznika walut stanowi układ, który umożliwia przeliczanie różnych miar na odpowiadające im wartości zapisane w innym systemie miar w stosunku do systemu pierwotnego. Projekt został wykonany w środowisku programowalnych układów logicznych[1] *Quartus II v5.1* firmy *Altera* z wykorzystaniem języka opisu sprzętu VHDL[2, 3] oraz docelowo zaimplementowany na płycie uruchomieniowej[4] *Trex C1* firmy *Terasic* w układzie[5] *FPGA Cyclone* EP1C6Q240C8.

2. PRZELICZNIK WALUT – OPIS PROJEKTU.

2.1. Założenia projektowe

Przy realizacji w/w projektu przyjęto następujące założenia projektowe:

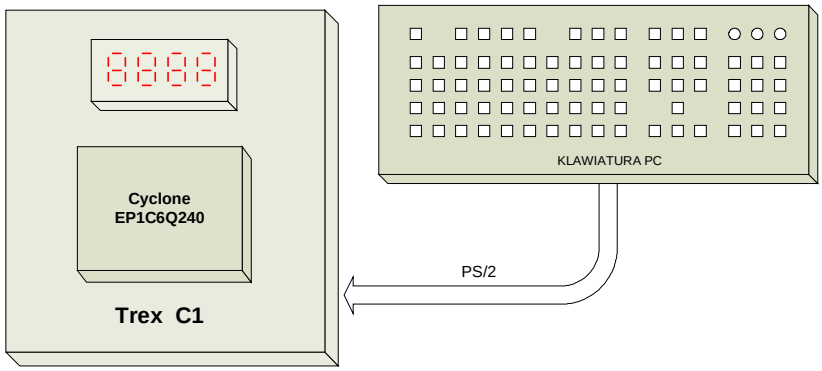
1. Komunikacja urządzenia z użytkownikiem będzie odbywać się za pomocą klawiatury PC z interfejsem *PS/2* oraz czteropozycyjnego, 7-segmentowego wyświetlacz LED znajdującego się na płycie uruchomieniowej.
2. Przyjęto, że klawisze funkcyjne *F1-F12* będą miały przypisane odpowiadające im wybrane jednostki miar (np. *dolar, euro*, itp.) wraz ze współczynnikami przeliczania w stosunku do innych systemów metrycznych - walut (np. *złotówka, jen*).
3. Maksymalna wprowadzona wartość (niezależnie od wybranej miary) będzie równa 100 – ze względu na ograniczenie ilości wyświetlanych cyfr (4).
4. Wynik przedstawiany będzie w postaci czterocyfrowej liczby na wyświetlaczu LED.

2.2. Struktura blokowa urządzenia

Ogólną strukturę blokową urządzenia przedstawiono na **rysunku 2.1**. Zasadniczo składa się ona z dwóch części:

- Klawiatury PC PS/2 – służy do komunikacji z użytkownikiem.
- Zestawu uruchomieniowego *Trex C1*.

Za pomocą klawiatury dokonywany jest wybór miar oraz wprowadzanie odpowiadających im wartości liczbowych. Zestaw uruchomieniowy zawiera układ programowalny z zaimplementowaną strukturą logiczna odpowiadającą za komunikację z klawiaturą, dokonywanie obliczeń, wyświetlanie aktualnego stanu, w jakim znajduje się urządzenie oraz wyświetlanie wyniku obliczeń.

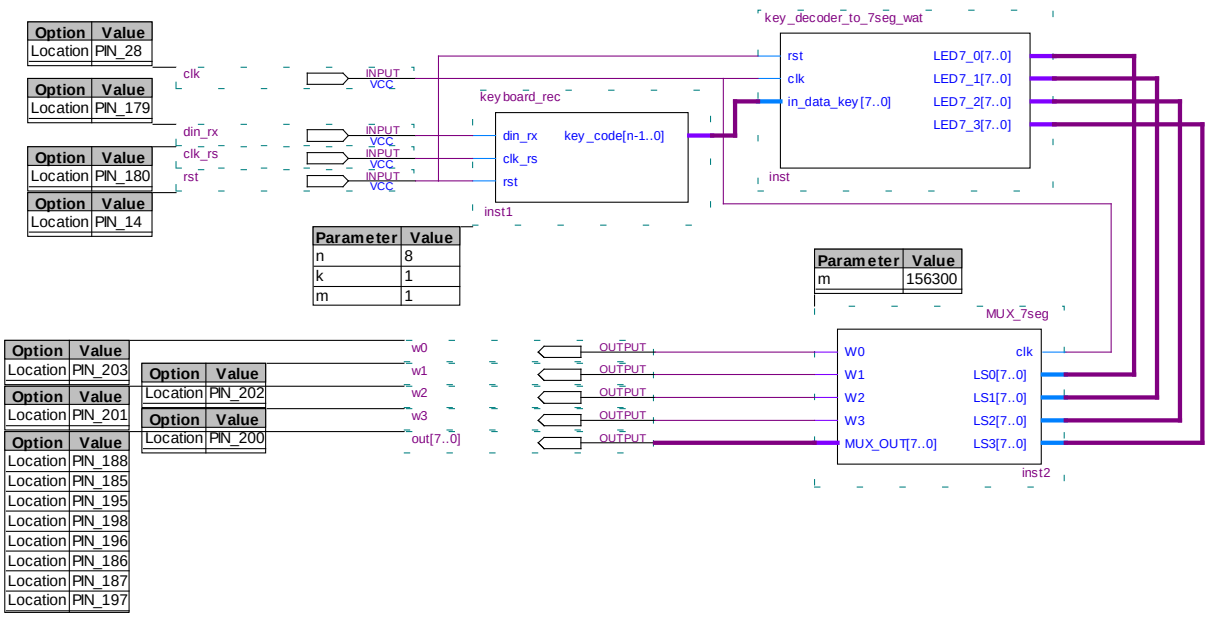


Rys. 2.1. Schemat blokowy.

2.3. Realizacja projektu w środowisku *Quartus*

Na **rysunku 2.2** przedstawiono blokową strukturę logiczna realizowanego projektu. Składa się ona z trzech zasadniczych części:

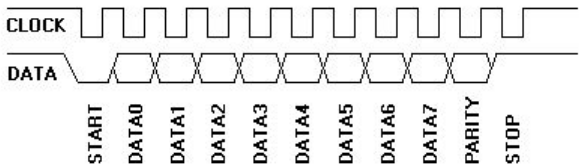
- Bloku *keyboard_rec* – odpowiedzialnego za komunikację z klawiaturą PS/2,
- Bloku *przelicznik* – odpowiedzialnego za interpretację kodu klawisza wysłanego z klawiatury PS/2, dokonywanie obliczeń oraz ustalanie stanów segmentów wyświetlacza LED.
- Bloku *mux_7seg* – bloku multiplexera sterującego pracą 4-pozycyjnego wyświetlacza LED.



Rys. 2.2. Blokowa struktura logiczna przelicznika walut.

2.3.1. Blok *keyboard_rec*

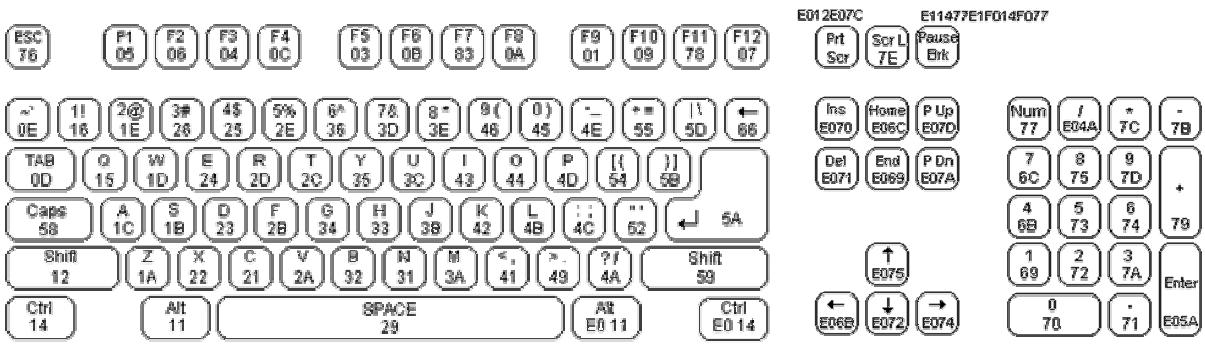
Blok ten odpowiedzialny jest za odbieranie danych z klawiatury PS/2. Komunikacja klawiatury z zestawem uruchomieniowym odbywa się za pośrednictwem synchronicznej transmisji szeregowej analogicznie ja w przypadku transmisji w standardzie RS232C, gdzie transmisja danych odbywa się z wykorzystaniem dwóch linii, z których pierwsza stanowi linię danych, a druga jest linią sygnału zegarowego. Typowa ramka danych[6] ma postać przedstawioną na **rysunku 2.3**. Składa się ona z 11 bitów, z których pierwszy jest bitem startu (wartość 0), kolejne 8 bitów jest kodem klawisza, następnym jest tzw. bit parzystości – równy 1 w przypadku nieparzystej liczby jedynek, a ostatnim bitem jest bit stopu – zawsze równy 1. Typowa wartość częstotliwości sygnału zegarowego mieści się w zakresie 10-16,7[kHz].



Rys. 2.3. Struktura ramki danych w transmisji z wykorzystaniem interfejsu PS/2.

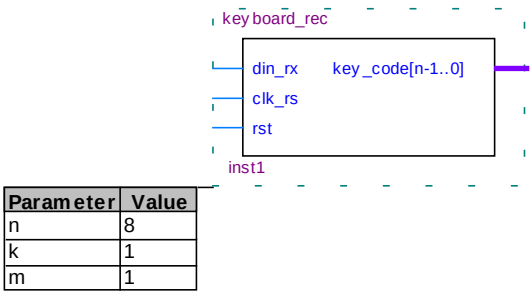
W momencie naciśnięcia klawisza na klawiaturze zostaje nadana ramka zawierająca w polu danych kod tego klawisza (na **rys. 2.4** przedstawiono mapę kodów klawiszy[7] typowej klawiatury z interfejsem PS/2) i jest ona nadawana aż do zwolnienia klawisza, natomiast w momencie zwolnienia klawisza zostaje nadany kod zwolnienia klawisza – F0 (szesnastkowo) + kod zwolnionego klawisza. Wynika z tego, że podczas pojedynczego cyklu naciśnięcia i zwolnienia klawisz nadawana jest sekwencja 3 bajtów – przykład poniżej:

- Naciśnięto klawisz ESC – nadany kod: 76h
- Zwolniono klawisz ESC – nadany kod: F0h, 76h.



Rys. 2.4. Mapa kodów klawiszy typowej klawiatury z interfejsem PS/2.

Na **rysunku 2.5** przedstawiono opisany językiem VHDL blok funkcyjny *keyboard_rec*. Opis tego bloku znajduje się z **załączniku nr 1**.



Parameter	Value
n	8
k	1
m	1

Rys. 2.5. Blok funkcyjny *keyboard_rec*.

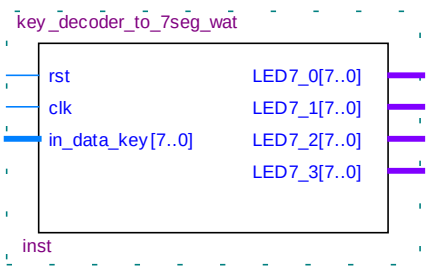
Od strony klawiatury do wejścia *din_rx* doprowadzony jest sygnał danych, a do wejścia *clk_rs* sygnał zegarowy. Odczyt danych odbywa się przy opadającym zboczu sygnału zegarowego na wejściu *clk_rs*. Parametry transmisji, tj. ilość bitów danych, obecność/brak bitu parzystości oraz bit stopu są ustalane za pomocą wartości parametrów *n, k, m*. Suma tych parametrów +1 stanowi o długości ramki danych. Pojedynczy cykl odbioru kodu klawisza wygląda następująco:

W momencie opadającego zbocza sygnału zegarowego na wejściu *clk_rs* uruchomiona zostaje procedura odbioru. Sprawdzany jest warunek czy najmłodsza pozycja w rejestrze danych jest równa 0 oraz czy linia danych *din_rx* ma stan odpowiadający logicznemu 0. Jeśli tak, to najmłodszy bit w rejestrze *reg* zostaje zapisany 1, co jest interpretowane, że rozpoczęto nadawanie ramki danych. Przy kolejnym opadającym zboczu zegara sprawdzany jest warunek czy *reg(0)=1* – jeśli tak to zostaje zwiększona o 1 wartość licznika danych *count*, po czym sprawdzany jest kolejny warunek czy wartość licznika *count* jest mniejsza od sumy parametrów *n+m+k*. Jeśli tak to pozycja w rejestrze *reg* o numerze równym aktualnej wartości licznika *count* zostaje zapisana wartością logiczną odpowiadającą aktualnemu stanowi na wejściu danych *din_rx*. Cykl ten powtarza się do momentu, w którym stan licznika jest równy *n+m+k*, co jest równoznaczne odebraniu całej ramki danych. Następnie wyzerowany zostaje licznik *count* oraz *bit(0)* w rejestrze *reg*. Wartość licznika ramek *count_L* jest zwiększana o 1. W kolejnym kroku sprawdzany jest warunek odnoszący się do aktualnej wartości licznika ramek *coun_L*. Jeśli wartość licznika jest równa 1, wówczas bity od n do 1 z rejestru *reg* zostają przepisane do rejestru ramki nr 1 *L1*, jeśli *count_L=2* , to do rejestru *L2*, a jeśli *count_L* jest równy 3 to dane przepisywane są do rejestru *L3* i wyzerowany zostaje licznik *count_L*. Następnie sprawdzane są warunki: czy któreś dwa z rejestrów *Lx* są sobie równe oraz czy trzeci z nich jest równy F0h – jeśli tak, to odebrano prawidłowo kod klawisza, a jego wartość zostaje przekazana na wyjście *key_code*, w przeciwnym razie stan na wyjściu

key_code nie ulega zmianie. Do bloku *keyboard_rec* do wejścia *rst* doprowadzony został również sygnał *resetu* (aktywny w stanie 0), który zeruje wszystkie rejestry oraz ustawia na wyjściu *key_code* stan FFh.

2.3.2. Blok *przelicznik*

Na **rysunku 2.6** przedstawiono zrealizowany blok przelicznik. Na wejście *in_data_key* wprowadzony zostaje kod odebranego klawisza. Zadaniem bloku *przelicznik* jest odpowiednie zinterpretowanie tego kodu i wykonanie przypisanej do niego funkcji. Zaimplementowano interpretację kodów następujących klawiszy: funkcyjnych *F1-F12*, zerowania *ESC*, *klawisz A* - pierwsza dana (miara), *klawisz B* druga dana (miara), *klawisz ENTER* – zatwierdzenie.



Rys. 2.6. Blok *przelicznik*.

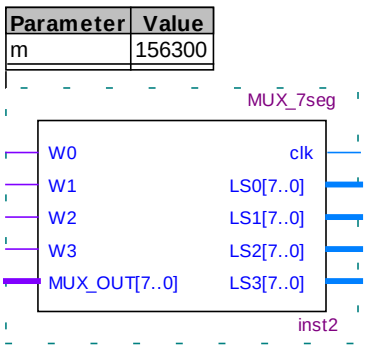
Według przyjętych założeń blok ten powinien działać następująco:

- W momencie uruchomienia projektu na wyświetlaczu pojawia się - - - -
- Naciskamy klawisz A – na wyświetlaczu pojawia się A- - -
- Następnie wybieramy walutę (miarę) za pomocą jednego z klawiszy F1 – F12 – na wyświetlaczu pojawia się np. A F1
- Naciskamy klawisz ENTER – na wyświetlaczu pojawia się A_ _ _ - w tym miejscu za pomocą klawiszy numerycznych powinna być wprowadzona wartość liczbowa od 0 do 100 - ta funkcja nie została zrealizowana.
- Następnie naciskamy klawisz B – na wyświetlaczu pojawia się B- - -
- Następnie wybieramy drugą walutę (miarę) za pomocą jednego z klawiszy funkcyjnych F1 – F12 – na wyświetlaczu pojawia się np. B F5
- Naciskamy klawisz ENTER – na wyświetlaczu powinien pojawić się wynik, niestety funkcja przeliczania miar nie została zaimplementowana, w wyniku, czego na wyświetlaczu pojawia się _ _ _ _
- Aby powrócić do stanu początkowego należy nacisnąć klawisz *ESC*.

Struktura logiczna bloku została opisana językiem VHDL i znajduje się w **załączniku nr 2**. Interpretacja kodu klawisza i wykonanie związanych z tym funkcji odbywa się w części synchronicznej opisanej procesem *decode*. Natomiast funkcje związane z obsługą odpowiednich segmentów wyświetlaczy LED są zawarte w części kombinacyjnej i zostały zrealizowane z wykorzystaniem polecenia *with … select*. Na poziomie tego bloku każdy wyświetlacz jest sterowany osobną 8-bitową szyną danych (*LED7_0…LED7_3*), co w dalszej części wymaga zastosowania bloku multipleksera – omówionego w punkcie 2.3.3.

2.3.3. Blok multipleksera *mux_7seg*

Na **rysunku 2.7** przedstawiono zrealizowany blok multipleksera. Opisany został za pomocą języka VHDL – opis znajduje się w **załączniku nr 3**. Opis ten można podzielić na dwie zasadnicze części, z których pierwsza zawiera opis dzielnika częstotliwości, natomiast druga zawiera właściwy opis multipleksera. Dzielnik częstotliwości zastosowano w celu spowolnienia pracy licznika sterującego multipleksowaniem poszczególnych wyświetlaczy, w taki sposób, by możliwy był jego odczyt. Było to niezbędne, ponieważ dostępny w zestawie uruchomieniowym sygnał zegarowy ma częstotliwość 50[MHz] (wprowadzany na wejście *clk*), a przy tej częstotliwości odczyt danych z wyświetlacza jest praktycznie niemożliwy. Uzyskana w efekcie podziału częstotliwości sygnału zegarowego nowa częstotliwość wynosi w przybliżeniu 400[Hz], co w efekcie daje około 80[Hz]/wyświetlacz. Do bloku multipleksera doprowadzone zostały szyny danych wyświetlaczy z bloku przelicznik. Wybór właściwej szyny danych (*LED7_0, …LED7_3*), oraz linii sterującej danym wyświetlaczem (*W0, … W3*) jest dokonywany na podstawie stanu licznika zliczającego od 0 do 3. Multipleksowane dane wprowadzane są na 8-bitową szynę danych *MUX_OUT*. Szczegółowy opis działania multipleksera został pokazany za pomocą przebiegów czasowych zawartych w **załączniku 3**.



Rys. 2.7. Blok multiplexera.

3. PODSUMOWANIE

Projekt nie został dokończony - brak jest zaimplementowanych w bloku *przelicznik* następujący funkcji:

1. Funkcja wprowadzania wartości liczbowej dla danej waluty (miary) – funkcja ta odpowiada za interpretację kodów klawiszy numerycznych (klawiatura PC) oraz wprowadzanie zinterpretowanej liczby (0-100) do wewnętrznego rejestru.
2. Przyporządkowanie do kodów klawiszy funkcyjnych F1-F12 współczynników przeliczania w zależności od wybranej waluty (miary).
3. Funkcja mnożąca – obliczająca iloczyn waluty A (o wartości 0-100) i waluty B.

Ze względu na próby realizacji w/w funkcji oraz niewłaściwe lub brak ich prawidłowego działania, fragmenty kodu w języku VHDL opisujące te funkcje zostały usunięte z listingu zamieszczonego w załączniku nr 2.

Pozostałe bloki zostały zrealizowane zgodnie z przyjętymi założeniami. Opis każdego z bloków znajduje się w odpowiednim załączniku wraz z przebiegami czasowymi utworzonymi na podstawie symulacji. Projekt został fizycznie uruchomiony z wykorzystaniem zestawu uruchomieniowego *Trex C1* i sprawdzony pod względem funkcjonalnym. Wszystkie funkcje związane z komunikacją z klawiaturą PC, interpretacją kodów klawiszy oraz prawidłowego wyświetlania komunikatów działają prawidłowo.

4. LITERATURA

[1] Altera Corporation: "*Quartus II Development Software Handbook v5.1*". Altera Corporation, San Jose 2005. http://www.altera.com/literature/lit-qts.jsp

[2] Zwoliński M.: „*Projektowanie układów cyfrowych z wykorzystaniem języka VHDL*", WKŁ, Warszawa 2002.

[3] Volnei A.: „*Circuit design with VHDL*". Massachusetts Institute of Technology, 2004.

[4] Terasic Technologies: „*Terasic TREX C1 Multimedia Development Kit*". 2005. http://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&CategoryNo=39

[5] Altera Corporation: " *Cyclone Device Handbook (All Sections) v2.0*". Altera Corporation.

[6] Computer-Engineering: „*The PS/2 Mouse/Keyboard Protocol*". http://www.computer-engineering.org/ps2protocol/

[7] Beyond Logic: „*Interfacing the AT keyboard*" http://www.beyondlogic.org/keyboard/keybrd.htm

5. WYKAZ ZAŁĄCZNIKÓW

[1] Załącznik Nr 1: „*Uniwersalny przelicznik walut – blok keyboard_rec*".

[2] Załącznik Nr 2: „*Uniwersalny przelicznik walut – blok przelicznik*".

[3] Załącznik Nr 3: „*Uniwersalny przelicznik walut – blok MUX_7seg*".

WOJSKOWA AKADEMIA TECHNICZNA
WYDZIAŁ ELEKTRONIKI

Laboratorium układów programowalnych	
Projekt w języku VHDL – ZAŁĄCZNIK Nr 1	
Temat projektu: **Uniwersalny przelicznik walut – blok *keyboard_rec***	
Grupa: **E6T1N2**	Data wykonania ćwiczenia: **14.10.2006**
Wykonał: **Jacek SANIEWSKI**	Ocena:
	Prowadzący: **dr inż. Z. JACHNA**

1. Opis linii wejścia – wyjścia (I/O)

Tab. 1. Opis linii I/O.

Nazwa	Typ	Opis
din_rx	I	Szeregowe wejście danych z interfejsu PS/2.
clk_rs	I	Sygnał zegarowy synchronizujący transmisję poprzez interfejs PS/2.
rst	I	Sygnał zerujący rejestry danych – aktywny w stanie ‘0’.
key_code	O	8-bitowy port wyjściowy. Na wyjście tego portu wprowadzany jest kod naciśniętego klawisza w zapisie szesnastkowym.

2. Opis bloku w języku VHDL

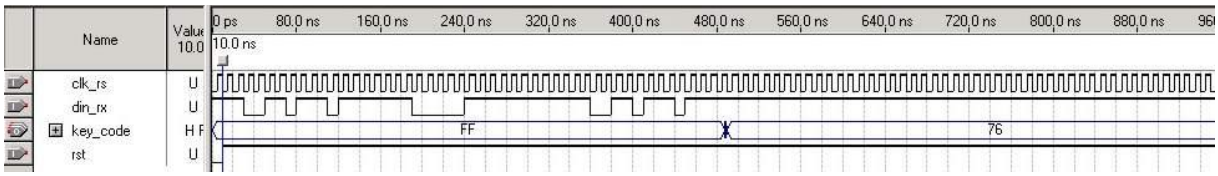
Tab. 2. Opis bloku w języku VHDL.

```
----------------------------------------------------------------------------------------------------------
-- BLOK ODBIORCZY W STANDARDZIE PS/2 - OBSŁUGA KLAWIATURY TYPU PS/2
-- Wersja V1.0
----------------------------------------------------------------------------------------------------------
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
-------------------------------------------------------------------------------
entity keyboard_rec is
generic (
                n: integer range 5 to 8:=8;             -- ilość bitów w słowie danych 5 do 8
                k: integer range 0 to 1:=1;             -- bit parzystości
                m: integer range 1 to 2:=1);            -- ilość bitów stopu 1 lub 2
-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
port(
        din_rx:             in      std_logic;
        clk_rs:             in      std_logic;
        rst:                in      std_logic;
        key_code:           out std_logic_vector ((n-1) downto 0)
       );
end keyboard_rec;
-------------------------------------------------------------------------------
architecture rtl of keyboard_rec is
begin
-=============================================================================
RECEIV: process (rst, clk_rs)

variable count: integer range 0 to (n+m+k+1);
variable count_L: integer range 0 to (n+m+k+1);
variable L1:  std_logic_vector ((n-1) downto 0);
variable L2:  std_logic_vector ((n-1) downto 0);
variable L3:  std_logic_vector ((n-1) downto 0);
variable reg: std_logic_vector ((n+m+k+1) downto 0);
begin
   ------------------------------------------------------
        ……..
        ……..
        ……..
        ……..
        ……..
        ……..
        ……..
        ……..
        ……..
        ……..
end process RECEIV;
-=============================================================================
end rtl;
```

3. Symulacja – przebiegi czasowe

Na poniższym rysunku przedstawiono symulację czasową bloku *keyboard_rec* w sytuacji naciśnięcia i zwolnienia klawisza *ESC*. Przebiegi czasowe odpowiadają sekwencji nadania następujących kodów:

1. Naciśnięcie klawisza ESC - kod 76h.
2. Zwolnienie klawisza ESC - kod F0h + 76h.



Rys. 1. Symulacja czasowa bloku *keyboard_rec* dla sekwencji wciśnięcia i zwolnienia klawisza ESC – kod 76h.

WOJSKOWA AKADEMIA TECHNICZNA
WYDZIAŁ ELEKTRONIKI

Laboratorium układów programowalnych	
Projekt w języku VHDL – ZAŁĄCZNIK Nr 2	
Temat projektu: **Uniwersalny przelicznik walut – blok *przelicznik***	
Grupa: **E6T1N2**	Data wykonania ćwiczenia: **14.10.2006**
Wykonał: **Jacek SANIEWSKI**	Ocena:
	Prowadzący: **dr inż. Z. JACHNA**

1. Opis linii wejścia – wyjścia (I/O)

Tab. 1. Opis linii I/O.

Nazwa	Typ	Opis
clk	I	Wejście sygnału zegarowego o częstotliwości 50[MHz].
rst	I	Wejście sygnału zerującego – aktywne w stanie ‘0’.
in_data_key	I	Wejściowa, 8-bitowa szyna danych – do tego wejścia przekazywany jest kod klawisza z bloku *keyboard_rec*.
led7_0	O	Wyjście sterujące 7-segmentowym wyświetlaczem LED.
led7_1	O	Wyjście sterujące 7-segmentowym wyświetlaczem LED.
led7_2	O	Wyjście sterujące 7-segmentowym wyświetlaczem LED.
led7_3	O	Wyjście sterujące 7-segmentowym wyświetlaczem LED.

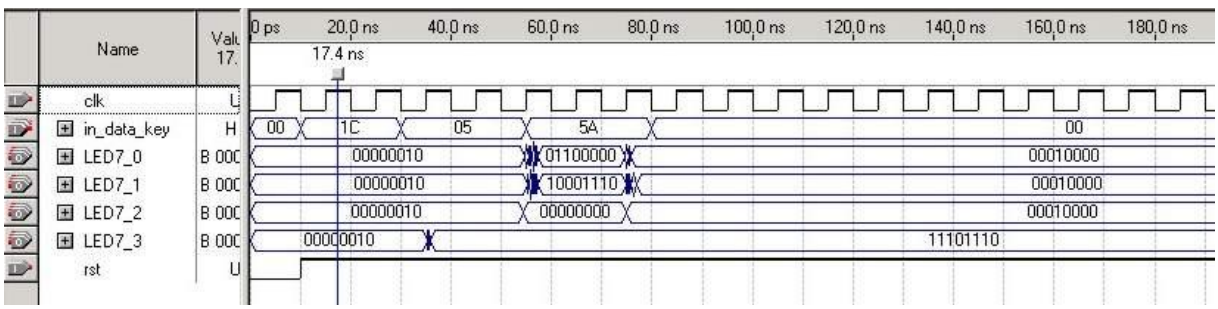
2. Opis bloku w języku VHDL

Tab. 2. Opis bloku w języku VHDL.

```
----------------------------------------------------------------------------
-- Blok przelicznik –
----------------------------------------------------------------------------

library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
----------------------------------------------------------------------------
entity przelicznik is
port(
        rst:                in      std_logic;
        clk:                in      std_logic;
        in_data_key:        in  std_logic_vector (7 downto 0);
        LED7_0:             out std_logic_vector (7 downto 0);
        LED7_1:             out std_logic_vector (7 downto 0);
        LED7_2:             out std_logic_vector (7 downto 0);
        LED7_3:             out std_logic_vector (7 downto 0)
       );
end przelicznik;
----------------------------------------------------------------------------
architecture rtl of przelicznik is
type key is (SA_F1, SA_F2, SA_F3, SA_F4, SA_F5, SA_F6,
                        SA_F7, SA_F8, SA_F9, SA_F10, SA_F11, SA_F12,
                        SB_F1, SB_F2, SB_F3, SB_F4, SB_F5, SB_F6,
                        SB_F7, SB_F8, SB_F9, SB_F10, SB_F11, SB_F12);
----------------------------------------------------------------------------
signal  LED7_s0:    std_logic_vector (7 downto 0):="10000111";
signal  LED7_s1:    std_logic_vector (7 downto 0):="10000111";
signal  LED7_s2:    std_logic_vector (7 downto 0):="10000111";
signal  LED7_s3:    std_logic_vector (7 downto 0):="10000111";

begin
--============================================================================
decode: process (rst, in_data_key, clk)

-------------------------------------------
begin
        ……..
        ……..
        ……..
        ……..
        ……..
----------------------------------------------------------------------------
end process decode;
--============================================================================
```

```
with LED7_s0 select

LED7_0 <=   --ABCDEFG.--
        ……..
        ……..
        ……..
                "00000000" when others;
-------------------------------------------------------
with LED7_s1 select

LED7_1 <=   --ABCDEFG.--
        ……..
        ……..
        ……..
                "00000000" when others;
-------------------------------------------------------
with LED7_s2 select

LED7_2 <=   --ABCDEFG.--
        ……..
        ……..
        ……..

                "00000000" when others;
-------------------------------------------------------
with LED7_s3 select

LED7_3 <=   --ABCDEFG.--
        ……..
        ……..
        ……..

                "00000000" when others;
-------------------------------------------------------------------------------
end rtl;
```

3. Symulacja – przebiegi czasowe



Rys. 1. Symulacja czasowa bloku *przelicznik* dla sekwencji kodów klawiszy: A, F1, ENTER.

WOJSKOWA AKADEMIA TECHNICZNA
WYDZIAŁ ELEKTRONIKI

Laboratorium układów programowalnych	
Projekt w języku VHDL – ZAŁĄCZNIK Nr 3	
Temat projektu: **Uniwersalny przelicznik walut – blok *MUX_7seg***	
Grupa: **E6T1N2**	Data wykonania ćwiczenia: **14.10.2006**
Wykonał: **Jacek SANIEWSKI**	Ocena:
	Prowadzący: **dr inż. Z. JACHNA**

1. Opis linii wejścia – wyjścia (I/O)

Tab. 1. Opis linii I/O.

Nazwa	Typ	Opis
clk	I	Wejście sygnału zegarowego o częstotliwości 50[MHz].
W0	O	Linia sterująca wyświetlaczem COM[0] (według dokumentacji zestawu Trex C1). Aktywna w stanie ‘0’.
W1	O	Linia sterująca wyświetlaczem COM[1] (według dokumentacji zestawu Trex C1). Aktywna w stanie ‘0’.
W2	O	Linia sterująca wyświetlaczem COM[2] (według dokumentacji zestawu Trex C1). Aktywna w stanie ‘0’.
W3	O	Linia sterująca wyświetlaczem COM[3] (według dokumentacji zestawu Trex C1). Aktywna w stanie ‘0’.
LS0	I	Wejście danych dla wyświetlacza COM[0].
LS1	I	Wejście danych dla wyświetlacza COM[1].
LS2	I	Wejście danych dla wyświetlacza COM[2].
LS3	I	Wejście danych dla wyświetlacza COM[3].
MUX_OUT	O	Wyjście dla wspólnej magistrali danych sterującej wyświetlaczami COM[0], … COM[3].

2. Opis bloku w języku VHDL

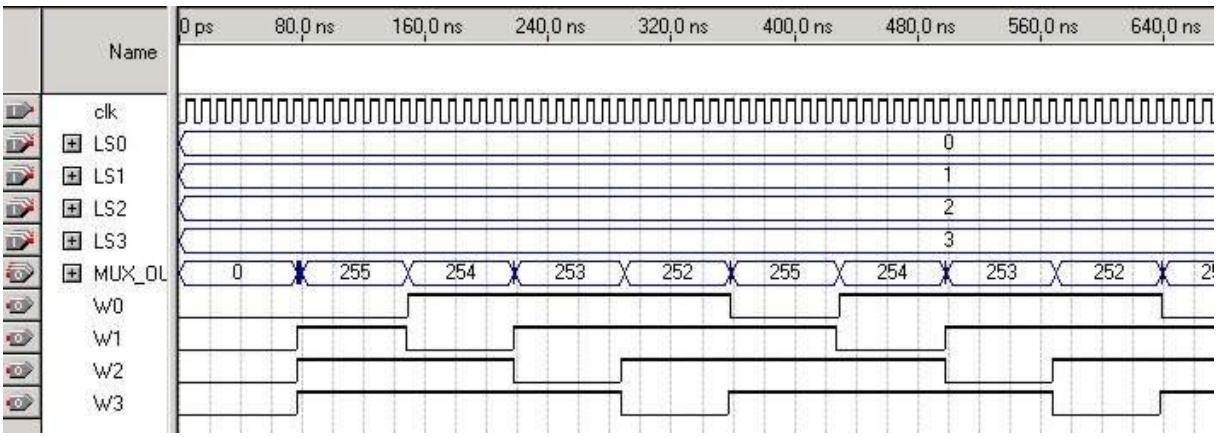
Tab. 2. Opis bloku w języku VHDL.

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
----------------------------------------------------
entity MUX_7seg is
generic ( m: integer:= 156250);
-- Parametr 'm' okresla do jakiej wartości ma zliczyć licznik,
-- aby uzyskać częstotliwość drugie sygnału zegarowego = 320Hz
-- f1=50MHz, T=0,00000002s
-- f=320Hz,  T=0,003125s   80Hz/wyświetlacz,
-- m=T2/T1
port(
        clk:            in      std_logic; -- zegar
        W0:             out     std_logic; -- sterowanie  LS0
        W1:             out     std_logic; -- sterowanie  LS1
        W2:             out     std_logic; -- sterowanie  LS2
        W3:             out     std_logic; -- sterowanie  LS3
        LS0:            in      std_logic_vector(7 downto 0); -- LED1
        LS1:            in      std_logic_vector(7 downto 0); -- LED2
        LS2:            in      std_logic_vector(7 downto 0); -- LED3
        LS3:            in      std_logic_vector(7 downto 0); -- LED4
        MUX_OUT:  out std_logic_vector(7 downto 0)  -- LED OUT
        );
end MUX_7seg;
----------------------------------------------------
architecture rtl of MUX_7seg is
signal Q:               std_logic_vector(1 downto 0):="00"; -- stan licznika
signal clk_l:           std_logic;      -- sygnał zegarowy z dzielnika
begin

--*****************************************************
-- DZIELNIK CZĘSTOTLIWOŚCI ---------------------------
--*****************************************************
fdiv: process (clk, clk_l) is
```

```
variable count_clk:          integer range 0 to m:=0;   -- rejestr licznika
begin
-------------------------------------------------------------
        ……..
        ……..
        ……..
end process fdiv;

--*****************************************************
-- LICZNIK ------------------------------------------
--*****************************************************

licznik: process (clk_l) is
begin
        ……..
        ……..
        ……..
end process licznik;

--*****************************************************
-- MULTIPLEXER ---------------------------------------
--*****************************************************
MUX: process ( clk_l, Q, LS0, LS1, LS2, LS3) is
begin
        ……..
        ……..
        ……..
end process MUX;
--*****************************************************
end rtl;
```

3. Symulacja – przebiegi czasowe

Symulacja czasowa na potrzeby pokazania pełnej sekwencji sterującej wyświetlaczami LED została przeprowadzona dla parametru m = 8, czyli częstotliwości sygnału zegarowego równej 6,25[MHz]. Wyjście *MUX_OUT* jest zanegowanym stanem jednego z wejść *LSx* aktywnego w danej chwili. Sytuacja taka ma miejsce, dlatego, że segmenty wyświetlaczy LED są sterowane stanem niskim, natomiast w bloku *przelicznik* przyjęto (dla zwiększenia czytelności opisu), że stanem aktywnym jest stan wysoki.



Rys. 1. Symulacja czasowa bloku *MUX_7seg*.