Katedra Informatyki Technicznej E-5

Architektura systemów komputerowych

Dokumentacja

Temat: 1 Port 8-bitowy z programowanymi wyjściami czasowymi o następujących funkcjach: generacja impulsu o ustalonym czasie trwania (1ms-10s) lub o ustalonej częstotliwości (1kHz-0.1Hz).

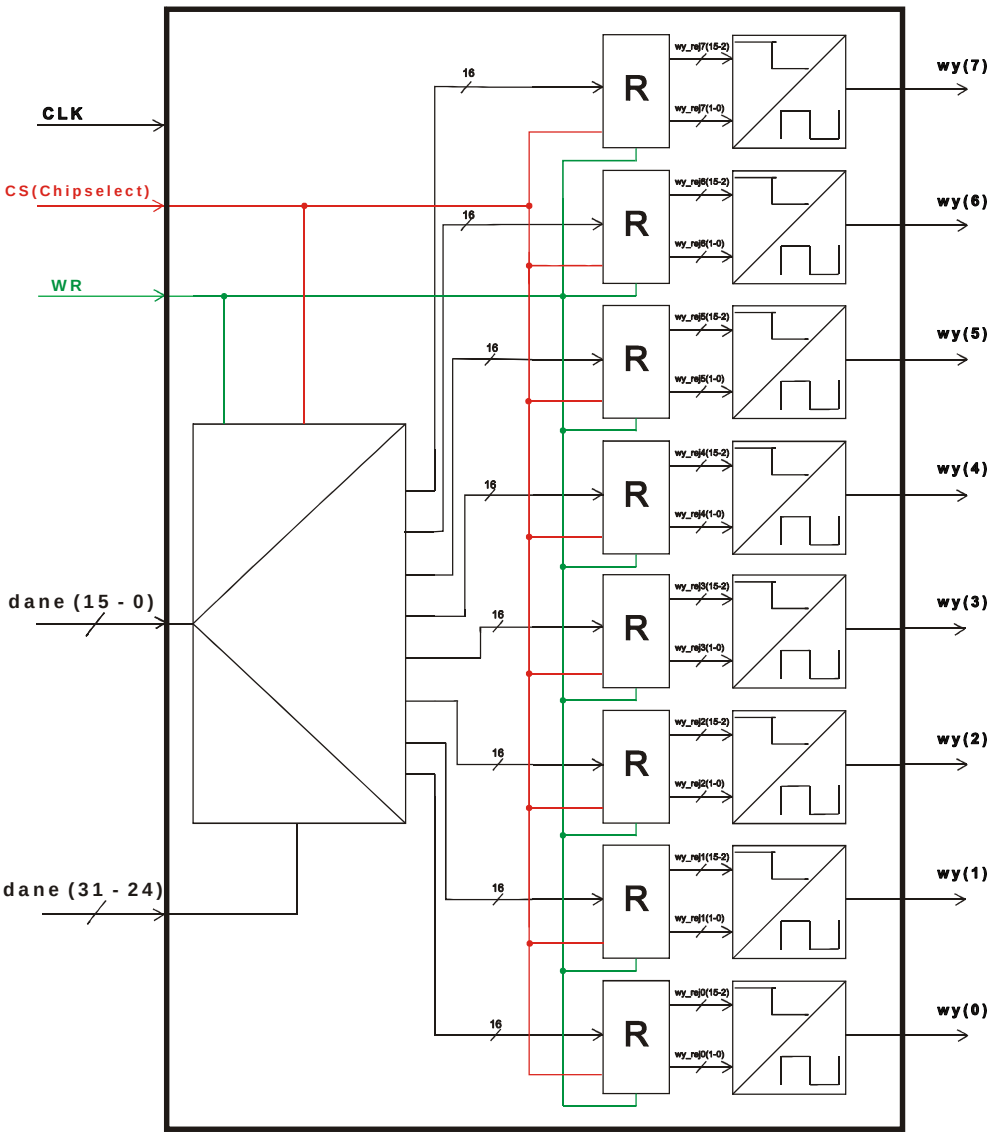
Wykonali:
Marcin Brach
GR. 25I P1

1. Cel projektu

Celem projektu jest zaprojektowanie i implementacja (w technologii FPGA z wykorzystaniem systemu prototypowego DE2, programu Quartus 10.0.SP1 /SOPC Builder) systemu komputerowego który składał się będzie z modułu procesora 32-bitowego Nios II interfejsu JTAG i modułu pamięci RAM oraz portu 8-bitowego z programowanymi wyjściami czasowymi o następujących funkcjach: generacja impulsu o ustalonym czasie trwania z zakresu 1ms - 10s lub o ustalonej częstotliwości z zakresu 1kHz – 0.1Hz. Moduł portu należy zaprojektować i zaimplementować w języku vhdl i dołączyć do systemu komputerowego, a następnie zaimplementować program testujący poprawność działania portu w języku C oraz przetestować go wykorzystując system prototypowy DE2.

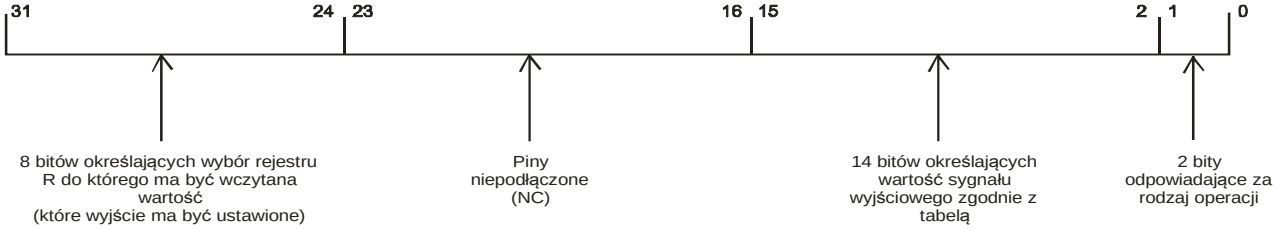
2. Sposób wykonania

Projektowanie modułu portu 8-bitowego odbywa się za pomocą programu Quartus 10.0.SP1. Poszczególne moduły dodawane są z bibliotek za pomocą narzędzia SOPC Builder, tylko własny moduł dodawany jest ręcznie przez dołączenie pliku vhdl z implementacją modułu.



Rysunek 1 Schemat blokowy projektowanego modułu

Na rysunku 1 przedstawiono schemat blokowy projektowanego modułu. Port składa się z ośmiu 16 bitowych rejestrów R do których przekazywana jest informacja z demultipleksera o rodzaju generowanego impulsu - *dane(1-0)*, i jego wartości - *dane(15-2)*. Na wyjściu każdego z rejestrów znajduje się układ generujący przebieg o określonej częstotliwości - ⎍ , lub o określonym czasie trwania - ⎍ wygenerowany przebieg znajduje się na wyjściu *wy(7-0)* w zależności od tego do którego rejestru została wczytana wartość o pożądanym przebiegu. Wybór tego rejestru odbywa się za pomocą wejścia *dane(31-24)* które stanowi wejście adresowe demultipleksera, rejestry są wybierane za pomocą kodu 1 z n (1 z 8). Cały układ jest wyzwalany poziomem wysokim, zapis do rejestrów jest możliwy tylko gdy sygnał zapisu *WR* i *CS (chipselect)* jest w poziomie wysokim.



Rysunek 2 Organizacja szyny wejściowej dane

Dane można wczytywać do rejestrów tylko za pomocą organizacji przedstawionej na rysunku 2.

Bity odpowiadające za rodzaj operacji to:

00 – na wyjściu jest poziom niski.

01 – generacja impulsu o określonym czasie trwania.

10 – generacja impulsu o określonej częstotliwości.

11 – na wyjściu jest poziom niski.

Na wyjściu układu można otrzymać przebieg o określonym czasie trwania z zakresu 1ms - 10s,
oraz przebieg o określonej częstotliwości z zakresu 1kHz-0.1Hz. Poniższa tabela przedstawia przykładowe wartości danych przesyłanych na szynę *dane(15-2)*:

Lp.	Wartości HEX:	Czas [s]:	Częstotliwość [Hz]:
1	1	0,001	1000
2	2	0,002	999,9
3	3	0,003	999,8
4	4	0,004	999,7
5	5	0,005	999,6
6	6	0,006	999,5
7	7	0,007	999,4
8	8	0,008	999,3
9	9	0,009	999,2
10	A	0,01	999,1
11	B	0,011	999
…	…	…	…
…	…	…	…
…	…	…	…
9989	2705	9,989	1,2
9990	2706	9,99	1,1
9991	2707	9,991	1
9992	2708	9,992	0,9
9993	2709	9,993	0,8
9994	270A	9,994	0,7
9995	270B	9,995	0,6
9996	270C	9,996	0,5
9997	270D	9,997	0,4
9998	270E	9,998	0,3
9999	270F	9,999	0,2
10000	2710	10	0,1

Wraz ze wzrostem podawanych wartości na *dane(15-2)* czas rośnie od 0,001s (1ms) do 10s, a częstotliwość maleje od 1000Hz (1kHz) do 0.1Hz.

3. Kod modułu w języku vhdl

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
entity uklad8 is                                                    --jednostka projektowa
port(           dane                    : in std_logic_vector(31 downto 0);
                        clk                     : in std_logic;
                        wr                      : in std_logic;
                        chipselect  : in std_logic;
                        wy                      : out std_logic_vector(7 downto 0)
                        );
end uklad8;
architecture RTL of uklad8 is
signal ms: std_logic;
signal w0: std_logic;                                               --deklaracja sygnałów
signal w1: std_logic;
signal w2: std_logic;
signal w3: std_logic;
signal w4: std_logic;
signal w5: std_logic;
signal w6: std_logic;
signal w7: std_logic;
signal wy_rej0: std_logic_vector(15 downto 0);
signal wy_rej1: std_logic_vector(15 downto 0);
signal wy_rej2: std_logic_vector(15 downto 0);
signal wy_rej3: std_logic_vector(15 downto 0);
signal wy_rej4: std_logic_vector(15 downto 0);
signal wy_rej5: std_logic_vector(15 downto 0);
signal wy_rej6: std_logic_vector(15 downto 0);
signal wy_rej7: std_logic_vector(15 downto 0);
Begin
process(clk)                                                        --proces zliczający do 1ms i ustawiający sygnał ms
variable x: integer range 0 to 50000;                               --ograniczenie zakresu od 0 do 50000 (1ms)
begin
        if(clk'event and clk='1') then
                ms<='0';
                x:=x+1;
        end if;
        if (x=50000) then
                x:=0;
                ms<='1';
        end if;
end process;
process(clk,wr,chipselect,dane,ms,w0,w1,w2,w3,w4,w5,w6,w7,wy_rej0,wy_rej1,wy_rej2,wy_rej3,wy_rej4,wy_rej5,wy_rej6,wy_rej7) is
variable rej0: std_logic_vector(15 downto 0);
variable rej1: std_logic_vector(15 downto 0);                       --deklaracja zmiennych  (rejestrów)
variable rej2: std_logic_vector(15 downto 0);
variable rej3: std_logic_vector(15 downto 0);
variable rej4: std_logic_vector(15 downto 0);
variable rej5: std_logic_vector(15 downto 0);
variable rej6: std_logic_vector(15 downto 0);
variable rej7: std_logic_vector(15 downto 0);
variable y: integer range 0 to 10000;
variable y1: integer range 0 to 10000;
variable y2: integer range 0 to 10000;                              --deklaracja zmiennych
variable y3: integer range 0 to 10000;                              --ograniczenie zakresu od 0 do 10000 (1ms -10s lub 1kHz-0.1Hz )
variable y4: integer range 0 to 10000;
variable y5: integer range 0 to 10000;
variable y6: integer range 0 to 10000;
variable y7: integer range 0 to 10000;
variable y8: integer range 0 to 10000;
variable y9: integer range 0 to 10000;
variable y10: integer range 0 to 10000;
variable y11: integer range 0 to 10000;
variable y12: integer range 0 to 10000;
variable y13: integer range 0 to 10000;
variable y14: integer range 0 to 10000;
variable y15: integer range 0 to 10000;
```

```
begin
        if (clk'event and clk='1'and wr='1' and chipselect='1') then                    --zapis danych do rejestrów
                case  dane(31 downto 24) is
                        when"00000001"=>   rej0 := dane(15 downto 0);
                        when"00000010"=>   rej1 := dane(15 downto 0);
                        when"00000100"=>   rej2 := dane(15 downto 0);
                        when"00001000"=>   rej3 := dane(15 downto 0);
                        when"00010000"=>   rej4 := dane(15 downto 0);
                        when"00100000"=>   rej5 := dane(15 downto 0);
                        when"01000000"=>   rej6 := dane(15 downto 0);
                        when"10000000"=>   rej7 := dane(15 downto 0);
                        when others   =>   null;
                        end case;
                end if;
                        wy_rej0 <= rej0;
                        wy_rej1 <= rej1;
                        wy_rej2 <= rej2;
                        wy_rej3 <= rej3;
                        wy_rej4 <= rej4;
                        wy_rej5 <= rej5;
                        wy_rej6 <= rej6;
                        wy_rej7 <= rej7;


        if(wy_rej0(1 downto 0)="00")then                                        --na wyjściu jest poziom niski
                wy(0)<='0';
        elsif(wy_rej0(1 downto 0)="01") then                                    --generacja impulsu o określonym czasie
                        if(clk'event and clk='1' and ms='1') then
                                y:=y+1;
                end if;
                if (conv_std_logic_vector(y,14)>=wy_rej0(15 downto 2)) then
                                w0 <='0';
                else
                                w0 <= '1';
                end if;
                wy(0)<=w0;
        elsif(wy_rej0(1 downto 0)="10") then                                    --generacja impulsu o określonej częstotliwości
                if(clk'event and clk='1' and ms='1') then
                                y1:=y1+1;
                if (conv_std_logic_vector(y1,14)=wy_rej0(15 downto 2)) then
                                w0 <= w0 xor '1';
                                y1:=0;
                end if;
        end if;
                wy(0)<=w0;                                                      --na wyjsciu jest poziom niski
        else
        wy(0)<='0';
        end if;

        if(wy_rej1(1 downto 0)="00")then
                        wy(1)<='0';
        elsif(wy_rej1(1 downto 0)="01") then
                if(clk'event and clk='1' and ms='1') then
                                y2:=y2+1;
                end if;
                if (conv_std_logic_vector(y2,14)>=wy_rej1(15 downto 2)) then
                                w1 <='0';
                else
                                w1 <= '1';
                end if;
                wy(1)<=w1;
        elsif(wy_rej1(1 downto 0)="10") then
                if(clk'event and clk='1' and ms='1') then
                                y3:=y3+1;
                if (conv_std_logic_vector(y3,14)=wy_rej1(15 downto 2)) then
                                w1 <= w1 xor '1';
                                y3:=0;
                end if;
                end if;
                wy(1)<=w1;
        else
        wy(1)<='0';
        end if;
```

```
if(wy_rej2(1 downto 0)="00")then
		wy(2)<='0';
elsif(wy_rej2(1 downto 0)="01") then
		if(clk'event and clk='1' and ms='1') then
					y4:=y4+1;
		end if;
		if (conv_std_logic_vector(y4,14)>=wy_rej2(15 downto 2)) then
					w2 <='0';
		else
					w2 <= '1';
		end if;
		wy(2)<=w2;
elsif(wy_rej2(1 downto 0)="10") then
		if(clk'event and clk='1' and ms='1') then
					y5:=y5+1;
		if (conv_std_logic_vector(y5,14)=wy_rej2(15 downto 2)) then
					w2 <= w2 xor '1';
					y5:=0;
		end if;
		end if;
		wy(2)<=w2;
else
wy(2)<='0';
end if;

if(wy_rej3(1 downto 0)="00")then
			wy(3)<='0';
elsif(wy_rej3(1 downto 0)="01") then
		if(clk'event and clk='1' and ms='1') then
					y6:=y6+1;
		end if;
		if (conv_std_logic_vector(y6,14)>=wy_rej3(15 downto 2)) then
					w3 <='0';
		else
					w3 <= '1';
		end if;
		wy(3)<=w3;
elsif(wy_rej3(1 downto 0)="10") then
		if(clk'event and clk='1' and ms='1') then
					y7:=y7+1;
		if (conv_std_logic_vector(y7,14)=wy_rej3(15 downto 2)) then
					w3 <= w3 xor '1';
					y7:=0;
		end if;
		end if;
		wy(3)<=w3;
else
wy(3)<='0';
end if;

if(wy_rej4(1 downto 0)="00")then
			wy(4)<='0';
elsif(wy_rej4(1 downto 0)="01") then
		if(clk'event and clk='1' and ms='1') then
					y8:=y8+1;
		end if;
		if (conv_std_logic_vector(y8,14)>=wy_rej4(15 downto 2)) then
					w4 <='0';
		else
					w4 <= '1';
		end if;
		wy(4)<=w4;
elsif(wy_rej4(1 downto 0)="10") then
		if(clk'event and clk='1' and ms='1') then
					y9:=y9+1;
		if (conv_std_logic_vector(y9,14)=wy_rej4(15 downto 2)) then
					w4 <= w4 xor '1';
					y9:=0;
		end if;
		end if;
		wy(4)<=w4;
else
wy(4)<='0';
end if;
```

```
	if(wy_rej5(1 downto 0)="00")then
				wy(5)<='0';
	elsif(wy_rej5(1 downto 0)="01") then
		if(clk'event and clk='1' and ms='1') then
					y10:=y10+1;
		end if;
		if (conv_std_logic_vector(y10,14)>=wy_rej5(15 downto 2)) then
					w5 <='0';
		else
					w5 <= '1';
		end if;
		wy(5)<=w5;
	elsif(wy_rej5(1 downto 0)="10") then
		if(clk'event and clk='1' and ms='1') then
					y11:=y11+1;
		if (conv_std_logic_vector(y11,14)=wy_rej5(15 downto 2)) then
					w5 <= w5 xor '1';
					y11:=0;
		end if;
		end if;
		wy(5)<=w5;
	else
	wy(5)<='0';
	end if;

	if(wy_rej6(1 downto 0)="00")then
		wy(6)<='0';
	elsif(wy_rej6(1 downto 0)="01") then
		if(clk'event and clk='1' and ms='1') then
					y12:=y12+1;
		end if;
		if (conv_std_logic_vector(y12,14)>=wy_rej6(15 downto 2)) then
					w6 <='0';
		else
					w6 <= '1';
		end if;
		wy(6)<=w6;
	elsif(wy_rej6(1 downto 0)="10") then
		if(clk'event and clk='1' and ms='1') then
					y13:=y13+1;
		if (conv_std_logic_vector(y13,14)=wy_rej6(15 downto 2)) then
					w6 <= w6 xor '1';
					y13:=0;
		end if;
		end if;
		wy(6)<=w6;
	else
	wy(6)<='0';
	end if;

	if(wy_rej7(1 downto 0)="00")then
		wy(7)<='0';
	elsif(wy_rej7(1 downto 0)="01") then
		if(clk'event and clk='1' and ms='1') then
					y14:=y14+1;
		end if;
		if (conv_std_logic_vector(y14,14)>=wy_rej7(15 downto 2)) then
					w7 <='0';
		else
					w7 <= '1';
		end if;
		wy(7)<=w7;
	elsif(wy_rej7(1 downto 0)="10") then
		if(clk'event and clk='1' and ms='1') then
					y15:=y15+1;
		if (conv_std_logic_vector(y15,14)=wy_rej7(15 downto 2)) then
					w7 <= w7 xor '1';
					y15:=0;
		end if;
		end if;
		wy(7)<=w7;
	else
	wy(7)<='0';
	end if;
end process;
end RTL;
```

4. Kod programu testującego w języku C

```
#include <stdio.h>
#include <io.h>

main()
{
                                                //generacja impulsu o określonym czasie trwania:
        IOWR(0x00000008,0,16777617);            //wczytanie do rejestru0 czasu 0,1s

        IOWR(0x00000008,0,33558433);            // wczytanie do rejestru1 czasu 1s

        IOWR(0x00000008,0,67128865);            // wczytanie do rejestru2 czasu 5s

        IOWR(0x00000008,0,134257729);           // wczytanie do rejestru3 czasu 10s
                                                //generacja impulsu o określonej częstotliwości:
        IOWR(0x00000008,0,268475422);           // wczytanie do rejestru4 częstotliwości 1Hz

        IOWR(0x00000008,0,536910518);           // wczytanie do rejestru4 częstotliwości 10Hz

        IOWR(0x00000008,0,1073777830);          // wczytanie do rejestru4 częstotliwości 100Hz

        IOWR(0x00000008,0,2147483654);          // wczytanie do rejestru4 częstotliwości 1000Hz


        printf("Wczytano do rejestrow!\n");
}
```