Ćwiczenie laboratoryjne 2

Celem ćwiczenia jest poznanie zasad funkcjonowania przerzutników i rejestrów.

Część I

Projektowanie przerzutników

W układach cyfrowych przerzutniki wykorzystywane są jako elementy zapamiętujące informację. Przerzutnik bistabilny może znajdować się w jednym z dwóch stanów. Klasyfikacja określająca sposób funkcjonowania przerzutników określa stany w jakich znajdują się wejścia i wyjścia przerzutnika w dwóch kolejnych taktach ts i ts+1. W praktyce wykorzystujemy następujące typy przerzutników: RS, D, JK, T, DV, E. Sposób funkcjonowania przerzutnika może być zadany tabelą przełączeń.

Zadania do wykonania

1. Zapoznać się z tabelami przełączeń podstawowych typów przerzutników.
2. Wykorzystując bramki NAND zbudować przerzutnik MS (typ przerzutnika określa prowadzący).
3. W języku VHDL opisać strukturalnie dany przerzutnik.
4. Opisać powyższe przerzutniki funkcjonalnie.
5. Sprawdzić działanie przerzutników w symulacji.

Część II

Projektowanie rejestrów

Celem ćwiczenia jest zapoznanie studentów z różnymi typami rejestrów , metodami ich projektowania. Zapoznanie się z instrukcjami VHDL opisującymi struktury regularne. Do realizacji ćwiczenia niezbędna jest znajomość teorii dotyczącej budowy i działania rejestrów. Operacje realizowane w projektowanych rejestrach:

- Y1 - ustawienie stanu początkowego rejestru (np. zerowanie)
- Y2 - zapis informacji do rejestru
- Y3 - iloczyn logiczny dwóch słów
- Y4 - suma logiczna dwóch słów
- Y5 - iloczyn logiczny dwóch słów
- Y6 - przesunięcie informacji w rejestrze (Y6l - w lewo , Y6p2 - w prawo o dwa bity),
- Y7 – inwertowanie zawartości rejestru

Dane do rejestru wprowadzamy z przełączników , wyjście rejestru na diody

Realizacja ćwiczenia polega na :

- przygotowanie elementów składowych (przerzutniki , zaprojektowanie rejestru jednobitowego – opisać go funkcjonalnie i strukturalnie)
- zaprojektowanie rejestru jednobitowego (opisać go funkcjonalnie i strukturalnie)
- na podstawie rejestru 1-bitowego zaprojektować rejestr n-bitowy zgodnie z tabelą wariantów
- sprawdzenie poprawności działania rejestrów za pomocą symulatora
- przydzielenie pinów wejściowych i wyjściowych
- zaprogramowanie układu
- prezentacja działającego układu

b3	b1	b0	typ przurzutnika	b2	b5	operacje	b4	b6	długośc rejestrów
0	0	0	T	0	0	y2-y4-y3	0	0	7
0	0	1	JK	0	1	y2-y3-y5	0	1	6
0	1	0	D	1	0	y2-y5-y1	1	0	5
0	1	1	RS	1	1	y2-y4-y5	1	1	8
1	0	0	JK	0	0	y2-y4-y3	0	0	7
1	0	1	T	0	1	y2-y3-y5	0	1	6
1	1	0	RS	1	0	y2-y5-y1	1	0	5
1	1	1	D	1	1	y2-y4-y5	1	1	8

Tabele wariantów realizowanych przez studentów

B3	B1	B0	długość rejestru i typ przerzutnika	B2	B5	B4	operacje
0	0	0	6 T	0	0	0	y2 y3 y4 y6l
0	0	1	8 JK	0	0	1	y2 y5 y4 y6p2
0	1	0	7 D	0	1	0	y2 y3 y5 y6l2
0	1	1	6 RS	0	1	1	y2 y5 y1 y6p
1	0	0	7 T	1	0	0	y2 y1 y4 y6p2
1	0	1	8 JK	1	0	1	y2 y3 y4 y6p
1	1	0	9 D	1	1	0	y2 y3 y5 y6l
1	1	1	9 JK	1	1	1	y2 y4 y5 y6l2

Przykład

Rejestr 8-bitowy-zerowanie,zapis,suma modulo2

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
entity rej8 is
        port    (       X:in STD_logic_vector(7 downto 0);
                        clk:in std_logic;
                        Y1:in std_logic;
                        Y2:in std_logic;
                        Y5:in std_logic;
                        out1:out std_logic_vector(7 downto 0));
end rej8;
architecture rej8b of rej8 is
component tr is                 -- komponent rejestr jednobitowy
        port    (               Y1:in std_logic;
                                Y2:in std_logic;
                                Y5:in std_logic;
                                clk:in std_logic;
                                X:in std_logic;
                                out1:out std_logic);
```

```
end component tr;
begin
		G_1:for i in 0 to 7 generate
			rej1:tr port map(Y1,Y2,Y5,clk,out1=>out1(i),X=>X(i));
		end generate G_1;
		end rej8b;
```

Komponenty rejestru

- **rejestr jednobitowy**

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
use work.all;
entity tr is
	port	(	X:in STD_logic;
			clk:in std_logic;
			Y1:in std_logic;
			Y2:in std_logic;
			Y5:in std_logic;
			out1:out std_logic);
end tr;
architecture rej of tr is
component przerzutnikT is	-- komponenty rejestru 1-bitowego/*
			port(		T:in std_logic;
					clk:in std_logic;
					Q:out std_logic;
					nQ:out std_logic);
end component przerzutnikT;
component not1 is
			port(		in1:in std_logic;
					y:out std_logic);
end component not1;
component or33 is
			port(		in1:in std_logic;
					in2:in std_logic;
					in3:in std_logic;
					y:out std_logic);
end component or33;

component and33 is
			port(		in1:in std_logic;
					in2:in std_logic;
					in3:in std_logic;
					y:out std_logic);
end component and33;

signal a11, a2,a21,a22,a3,TT,nQ,nX,outt1:std_logic;
begin
		out1<=outt1;
		nn:not1 port map(in1=>X,y=>nX);
```

```
		--operacja resetY1
		b1:and33 port map(in1=>y1,in2=>outt1,in3=>y1,y=>a11);
		--operacja zapisY2
		b2:and33 port map(in1=>Y2,in2=>nX,in3=>outt1,y=>a2);
		b21:and33 port map(in1=>Y2,in2=>X,in3=>nQ,y=>a21);
		n2:or33 port map(in1=>a2,in2=>a21,in3=>a2,y=>a22);
		--operacja moduloY5
		b3:and33 port map(in1=>Y5,in2=>X,in3=>x,y=>a3);
		ns2:or33 port map(in1=>a11,in2=>a22,in3=>a3,y=>TT);
		p1:przerzutnikT port map(clk=>clk,T=>TT,Q=>outt1,nQ=>nQ);
end rej;
```

- **Przerzutnik T**

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
entity przerzutnikT is
	port(	t : in STD_LOGIC;
		clk : in STD_LOGIC;
		Q : out STD_LOGIC;
		nQ : out STD_LOGIC
	);
end przerzutnikT;

architecture przerzutnikT of przerzutnikT is
begin

	xx:process(clk,t)
	variable QQ :std_logic:='1';
	variable NQQ : std_logic:='0';
begin
	if clk= '0' and clk'event then
		if t='1' then QQ:= not QQ;
			NQQ:= not QQ;
		end if;
	end if;
	Q<=QQ;
	NQ<=NQQ	;
end process xx;

end przerzutnikT;
```