UNIWERSYTET ZIELONOGÓRSKI
INSTYTUT INFORMATYKI I ELEKTROTECHNIKI
ZAKŁAD INŻYNIERII KOMPUTEROWEJ

Przygotował: dr inż. Remigiusz Wiśniewski

LABORATORIUM 3: MODELOWANIE PROCEDURALNE (1)

Cel ćwiczenia

Celem ćwiczenia jest zapoznanie się z możliwościami tworzenia i wykorzystywania modeli na poziomie funkcjonalnym. Instrukcje *always, initial*, dyrektywa *`timescale*.

Zadanie 1. Przygotować opis bramki *XOR*, napisać moduł testujący napisaną bramkę, a następnie przeprowadzić symulację układu. Zarówno model bramki, jak i moduł testujący powinien zawierać instrukcje proceduralne (*always, initial*). Wykorzystać dyrektywę *`timescale*.

- Przykład realizacji bramki Xor:

```
module Bramka_Xor(Y, A, B);
      input A,B;
      output Y;
      reg Y;

      always @(A,B) Y=A^B;
endmodule
```

- Przykład modułu testującego:

```
module Test;
      reg A;
      reg B;
      wire Y;

      BramkaXor U1 (Y,A,B);

      initial
      begin
            $monitor($realtime,," A=%b B=%b C=%b ",A,B,Y);
            A=1'b0;
            B=1'b0;
            #100
            A=1'b0;
            B=1'b1;
            #100
            A=1'b1;
            B=1'b0;
            #100
            A=1'b1;
            B=1'b1;
            #100
            $finish;
   end
endmodule
```

Zadanie 2. **Przygotować model przerzutnika typu D, aktywnego narastającym zboczem zegara. Przerzutnik powinien posiadać dwa wejścia: informacyjne *D* oraz zegarowe *Clk*, a także jedno wyjście *Q*. Następnie napisać moduł testujący i przeprowadzić symulację układu.**

- Narastające zbocze sygnału to ***posedge***, opadające zaś to ***negedge***.
- W modelu testującym należy wykorzystać generator zegara (instrukcja *always*).
  Przykład generatora sygnału zegarowego *Clk*:

```
always
begin
      #50 Clk=0;
      #50 Clk=1;
end
```

Zadanie 3. **Zmodyfikować model przerzutnika z zadania 2 dodając synchroniczne wejście ustawiające oraz asynchroniczne wejście zerujące.**

Zadanie 4. **Wykorzystując przypisania proceduralne przygotować opis multipleksera 2-bitowego *Mux_4* (dwa wejścia adresowe, cztery wejścia informacyjne, jedno wyjście). Do realizacji układu wykorzystać instrukcję *if*. Wejście adresowe powinno być wektorem. Sprawdzić różnice w zapisie warunku:**

```
if (Sel[0] == 1'b0 && Sel[1] == 1'b0) Y=A;
```

oraz:

```
if (Sel == 0) Y=A;
```

Zadanie 5. **Wykorzystując przypisania proceduralne przygotować opis multipleksera 2-bitowego *Mux_4* (dwa wejścia adresowe, cztery wejścia informacyjne, jedno wyjście). Do realizacji układu wykorzystać instrukcję *case*. Wejścia informacyjne powinny być 4-bitowymi wektorami.**