Sprawozdanie

Język opisu sprzętu (HDL)

Ćwiczenie 2: Układ arytmetyczno logiczny

Wykonane przez	
Imię i Nazwisko	Indeks
Miłosz Stasiak	240471
Filip Grzymski	240410

Politechnika Łódzka – 26 listopada 2023

SPIS TREŚCI

1 Treść zadania

Utworzyć moduł z czterema sygnałami wejściowymi: 8-bitową liczbą A, 8-bitową liczbą B, dwubitowym wyboru operacji oraz bitem C (przeniesienie/pożyczka) oraz z dwoma sygnałami wyjściowymi: 8-bitową liczbą wyniku oraz 4-bitowym słowem statusowym w, którym mają się znaleźć bity: C- (przeniesienie/pożyczka), bit parzystości typu EVEN wyniku, bit Z - znacznik wartości zero, bit-OV overflow określający przekroczenie zakresu dla arytmetyki liczb ze znakiem.

Zaprojektować cztery podukłady, dwa w formie TASK-u , mają one stanowić iteracyjny blok dla poszczególnych zadań, a dwa w formie wyrażeń logicznych:

1. arytmetyczna suma z przeniesieniem (wejścia: przeniesienie, bit liczby A, bit liczby B, wyjścia: bit sumy, bit przeniesienia)
2. arytmetyczna różnica $(A - B)$ z pożyczką (wejścia: pożyczka, bit liczby A, bit liczby B, wyjścia: bit sumy, bit pożyczki)
3. logiczna suma dwóch liczb 8-bitowych
4. logiczny iloczyn dwóch liczb 8-bitowych

Następnie zaprojektować układ wykorzystując utworzone wcześniej podukłady.

Układ w zależności od sygnału wyboru 1=0, ma realizować zadania arytmetyczne (wybor0=zero) sumę liczb z bitem C jako przeniesienie lub (wybor 0 = jeden) różnicę liczb (A-B) z bitem C jako pożyczka. Uzupełnić układ elementami kombinacyjnym wyznaczającymi sygnały Z i OV oraz bit parzystości typu EVEN.

Układ dla sygnału wyboru $1 = 1$, ma realizować zadania logiczne (wybor0=zero) sumę logiczną wyrażeń, (wybor0=jeden) iloczyn logiczny wyrażeń. Uzupełnić układ elementami kombinacyjnym wyznaczającymi sygnały Z oraz bit parzystości typu EVEN. Pozostałe dwa bity wynikowego słowa statusowego należy ustawić na zero.

2 Arytmetyczna suma i różnica dwóch liczb

Do działań arytmetycznych stworzyliśmy osobny ”task”, przy pomocy którego obliczane jest równanie, z elementem przeniesienia/pożyczenia

```
task sumator(input la, lb, p, output wynik, prze);
    begin
        prze = (la & lb) | (la & p) | (p & lb);
        wynik = (la ^ lb) ^ p;
    end
endtask

task roznica(input la, lb, p, output wynik, prze);
    begin
        prze = (~la & lb) | (lb & p) | (~la & p);
        wynik = (~la & ~lb & p) | (~la & lb & ~p) | (la & ~lb & ~p) | (la & lb & p);
    end
endtask
```

3 Logiczna suma i iloczyn dwóch liczb

Z powodu tego jak funkcjonuje oprogramowanie, wystarczyła jednolinikowa definicja logiki dla obydwu przypatków. Została ona umieszczona w funkcji wyboru, przełączanej później switch’ami.

```
    else if(wybor == 2'b10)
    begin
        wynik = liczbaA | liczbaB;
    end

    else if(wybor == 2'b11)
    begin
        wynik = liczbaA & liczbaB;
    end
```

4 Całość kodu

```
`timescale 1ns / 1ps


module main(
    input [7:0] liczbaA,
    input [7:0] liczbaB,
```

```
     input [1:0] wybor,
     input bitP,
     output reg[7:0] wynik,
     output C,
     output EVEN,
     output Z,
     output OV
    );

reg [7:0] pw;
reg z1;
reg z2;
reg zw;
reg ov;

task sumator(input la, lb, p, output wynik, prze);
    begin
        prze = (la & lb) | (la & p) | (p & lb);
        wynik = (la ^ lb) ^ p;
    end
endtask

task roznica(input la, lb, p, output wynik, prze);
    begin
        prze = (~la & lb) | (lb & p) | (~la & p);
        wynik = (~la & ~lb & p) | (~la & lb & ~p) | (la & ~lb & ~p) | (la & lb & p);
    end
endtask

always @(wybor or liczbaB or liczbaA or bitP)
begin
    pw = 0;
    ov = 0;

    if (wybor == 2'b00)
    begin
        z1 = liczbaA[7];
        z2 = liczbaB[7];

        sumator(liczbaA[0],liczbaB[0],bitP,wynik[0], pw[0]);
        sumator(liczbaA[1],liczbaB[1],pw[0],wynik[1], pw[1]);
        sumator(liczbaA[2],liczbaB[2],pw[1],wynik[2], pw[2]);
        sumator(liczbaA[3],liczbaB[3],pw[2],wynik[3], pw[3]);
        sumator(liczbaA[4],liczbaB[4],pw[3],wynik[4], pw[4]);
        sumator(liczbaA[5],liczbaB[5],pw[4],wynik[5], pw[5]);
        sumator(liczbaA[6],liczbaB[6],pw[5],wynik[6], pw[6]);
        sumator(liczbaA[7],liczbaB[7],pw[6],wynik[7], pw[7]);

        zw = wynik[7];
        ov = (~z1 & ~z2 & zw) | (z1 & z2 & ~zw);
```

```
    end

    else if(wybor == 2'b01)
    begin
        z1 = liczbaA[7];
        z2 = liczbaB[7];

        roznica(liczbaA[0],liczbaB[0],bitP,wynik[0], pw[0]);
        roznica(liczbaA[1],liczbaB[1],pw[0],wynik[1], pw[1]);
        roznica(liczbaA[2],liczbaB[2],pw[1],wynik[2], pw[2]);
        roznica(liczbaA[3],liczbaB[3],pw[2],wynik[3], pw[3]);
        roznica(liczbaA[4],liczbaB[4],pw[3],wynik[4], pw[4]);
        roznica(liczbaA[5],liczbaB[5],pw[4],wynik[5], pw[5]);
        roznica(liczbaA[6],liczbaB[6],pw[5],wynik[6], pw[6]);
        roznica(liczbaA[7],liczbaB[7],pw[6],wynik[7], pw[7]);

        zw = wynik[7];
        ov = (~z1 & z2 & zw) | (z1 & ~z2 & zw);
    end

    else if(wybor == 2'b10)
    begin
        wynik = liczbaA | liczbaB;
    end

    else if(wybor == 2'b11)
    begin
        wynik = liczbaA & liczbaB;
    end
end


assign C = pw[7];
assign EVEN = ^wynik;
assign Z = (wynik == 0);
assign OV = ov;

endmodule
```

5 przypisanie przycisków

```
NET "bitP" LOC = "p94" ;
NET "C" LOC = "p104" ;
NET "EVEN" LOC = "p102" ;
NET "liczbaA<0>" LOC = "p2" ;
```

```
NET "liczbaA<1>" LOC = "p4" ;
NET "liczbaA<2>" LOC = "p6" ;
NET "liczbaA<3>" LOC = "p9" ;
NET "liczbaA<4>" LOC = "p3" ;
NET "liczbaA<5>" LOC = "p5" ;
NET "liczbaA<6>" LOC = "p7" ;
NET "liczbaA<7>" LOC = "p10" ;
NET "liczbaB<0>" LOC = "p133" ;
NET "liczbaB<1>" LOC = "p135" ;
NET "liczbaB<2>" LOC = "p138" ;
NET "liczbaB<3>" LOC = "p140" ;
NET "liczbaB<4>" LOC = "p134" ;
NET "liczbaB<5>" LOC = "p136" ;
NET "liczbaB<6>" LOC = "p139" ;
NET "liczbaB<7>" LOC = "p142" ;
NET "OV" LOC = "p97" ;
NET "wybor<0>" LOC = "p124" ;
NET "wybor<1>" LOC = "p39" ;
NET "wynik<0>" LOC = "p112" ;
NET "wynik<1>" LOC = "p114" ;
NET "wynik<2>" LOC = "p116" ;
NET "wynik<3>" LOC = "p118" ;
NET "wynik<4>" LOC = "p113" ;
NET "wynik<5>" LOC = "p115" ;
NET "wynik<6>" LOC = "p117" ;
NET "wynik<7>" LOC = "p119" ;
NET "Z" LOC = "p100" ;
```