САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ

*Кафедра Вычислительной техники*

*Цифровая схемотехника*

**Лабораторная работа №1**

Вариант 3

Выполнил:

студент III курса группы 2125

Припадчев Артём

Преподаватель:

Попов Р.И.

Санкт-Петербург

2014

**Цель работы:**

1. Изучение основ языка Verilog HDL
2. Освоение принципов работы в среде Xilinx ISE

**Задание:**

Построить многотактный 8-разрядный целочисленный умножитель. При реализации не допускается использование умножителей. (Простейший возможный алгоритм - умножение “столбиком”, допускается использование другого алгоритма на ваш выбор)

Интерфейс модуля:

|  |  |  |  |
| --- | --- | --- | --- |
| Имя | Направление | Ширина | Описание |
| clk | input | 1 | Сигнал синхронизации |
| rstn | input | 1 | Синхронный сброс по уровню лог. 0 |
| a\_in | input | 8 | Первый операнд |
| b\_in | input | 8 | Второй операнд |
| start | input | 1 | стартовый импульс (запуск вычислений) |
| data\_out | output | 16 | Результат умножения |
| ready | output | 1 | Признак завершения вычислений, сигнализирует что на data\_out установлен результат умножения |

**Реализация:**

**Умножитель**

`timescale 1ns / 1ps

module top(

 input clk,

 input rstn,

 input wire [7:0] a\_in,

 input wire [7:0] b\_in,

 input wire start,

 output ready,

 output [14:0] data\_out

 );

reg [14:0] mult\_reg;

reg mult\_ready;

reg [6:0] res\_and;

reg [3:0] count;

reg sign;

reg temp;

assign data\_out = mult\_reg;

assign ready = mult\_ready;

always @(posedge clk or negedge rstn) begin

 if(~rstn)

 begin

 mult\_reg = 0;

 mult\_ready = 0;

 count = 0;

 sign = 0;

 end

 else

 begin

 if(start)

 begin

 if(a\_in[7] != b\_in[7])

 begin

 sign = 1;

 end

 temp = b\_in[count];

 //res\_and = a\_in & b\_in[count];

 if(b\_in[count])

 res\_and = a\_in[6:0];

 else

 res\_and = 0;

 count = count + 1;

 mult\_reg [14:7] = mult\_reg[14:7] + res\_and;

 mult\_reg = mult\_reg >> 1;

 if (count == 7)

 begin

 mult\_reg[14] = sign;

 mult\_ready = 1;

 end

 end

 end

 end

endmodule

**Тест**

`timescale 10ns / 1ps

module top\_tb;

 // Inputs

 reg clk;

 reg rstn;

 reg [7:0] a\_in;

 reg [7:0] b\_in;

 reg start;

 // Outputs

 wire ready;

 wire [14:0] data\_out;

 // Instantiate the Unit Under Test (UUT)

 top uut (

 .clk(clk),

 .rstn(rstn),

 .a\_in(a\_in),

 .b\_in(b\_in),

 .start(start),

 .ready(ready),

 .data\_out(data\_out)

 );

 initial begin

 clk = 0;

 forever #0.1 clk = ~clk;

 end

 initial begin

 rstn = 0;

 start = 0;

 @(posedge clk);

 @(posedge clk);

 rstn = 1;

 end

 integer i;

 initial begin

 @(posedge rstn);

 @(posedge clk);

 a\_in = 64;

 a\_in[7]=1;

 b\_in = 64;

 b\_in[7]=1;

 for(i=0; i<=126; i = i + 1)

 begin

 start = 1;

 $display("i = %d", i);

 if(a\_in[7]==0)

 $display("a\_in = %d", a\_in[7:0]);

 else

 $display("a\_in = -%d", a\_in[6:0]);

 if(b\_in[7]==0)

 $display("b\_in = %d", b\_in[7:0]);

 else

 $display("b\_in = -%d", b\_in[6:0]);

 while(~ready)

 begin

 $display("data\_out = %d", data\_out[13:0]);

 @(posedge clk);

 end

 start = 0;

 if(data\_out[14]==0)

 $display("data\_out = %d", data\_out[14:0]);

 else

 $display("data\_out = -%d", data\_out[13:0]);

 $display("================================");

 $display("");

 rstn = 0;

 a\_in = a\_in + 1;

 b\_in = b\_in - 1;

 @(posedge clk);

 rstn = 1;

 end

 $stop();

 end

endmodule

**Вывод:** в процессе работы были изучены основы языка Verilog HDL и рассмотрен принцип работы в среде Xilinx ISE

