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;
```

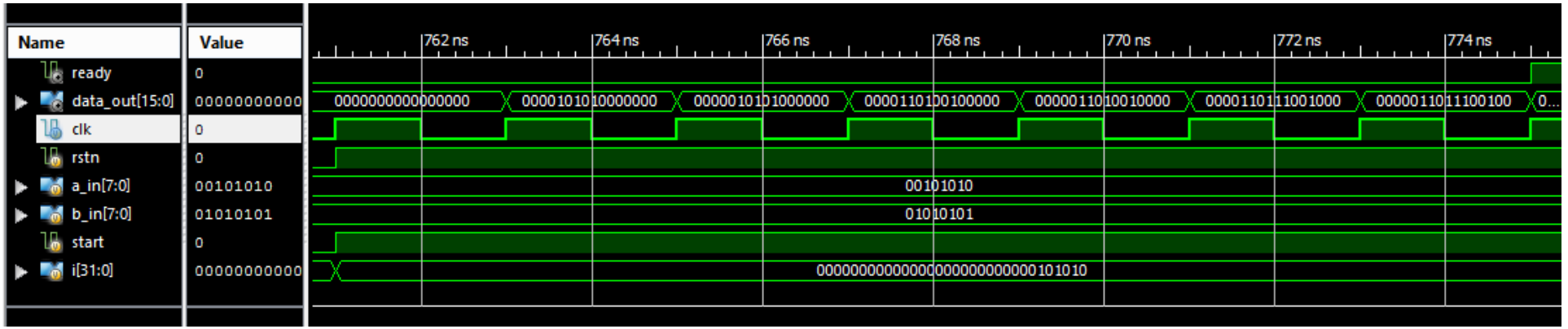
```
                                mult_reg [14:7] = mult_reg[14:7] + res_and;
                                mult_reg = mult_reg >> 1;
                                if (count == 7)
                                begin
                                        mult_reg[14] = sign;
                                        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


Name
Value
ready
0
data_out[15:0]
0000000000000000
clk
0
rstn
0
a_in[7:0]
00101010
b_in[7:0]
01010101
start
0
i[31:0]
00000000000
762 ns
764 ns
766 ns
768 ns
770 ns
772 ns
774 ns
0000000000000000
0000101010000000
0000010101000000
0000110100100000
0000110100010000
0000110111001000
0000110111100100
00101010
01010101
00000000000000000000000000101010