Федеральное государственное автономное образовательное учреждение
высшего
образования «Национальный исследовательский университет ИТМО»

Факультет программной инженерии и компьютерной техники

Отчет

По лабораторной работе №2

По дисциплине «Основы схемотехники»

Вариант 6.

Выполнил: Чураков Александр Алексеевич,
группа P3331
Преподаватель: Лукашов И. В.

Санкт-Петербург
2025

Оглавление

Цель работы 3
Задание 3
Отчет о проделанной работе 4
Область допустимых значений 4
Умножитель 16x8 4
Функция $ab+a^3$ 6
Описание работы 6
Тестовое окружение 8
Результат тестирования 10
Выводы 11

Цель работы

Получить навыки описания арифметических блоков на RTL-уровне с использованием языка описания аппаратуры Verilog HDL.

Задание

6	$y = a \cdot b + a^3$	2 сумматора и 1 умножитель

1. Разработайте и опишите на Verilog HDL схему, вычисляющую значение функции в соответствии с заданными ограничениями согласно варианту задания.

2. Определите область допустимых значений функции.

3. Разработайте тестовое окружение для разработанной схемы. Тестовое окружение должно проверять работу схемы не менее, чем на 10 различных тестовых векторах.

4. Проведите моделирование работы схемы и определите время вычисления результата. Схема должна тактироваться от сигнала с частотой 100 МГц.

5. Составьте отчет по результатам выполнения работы.

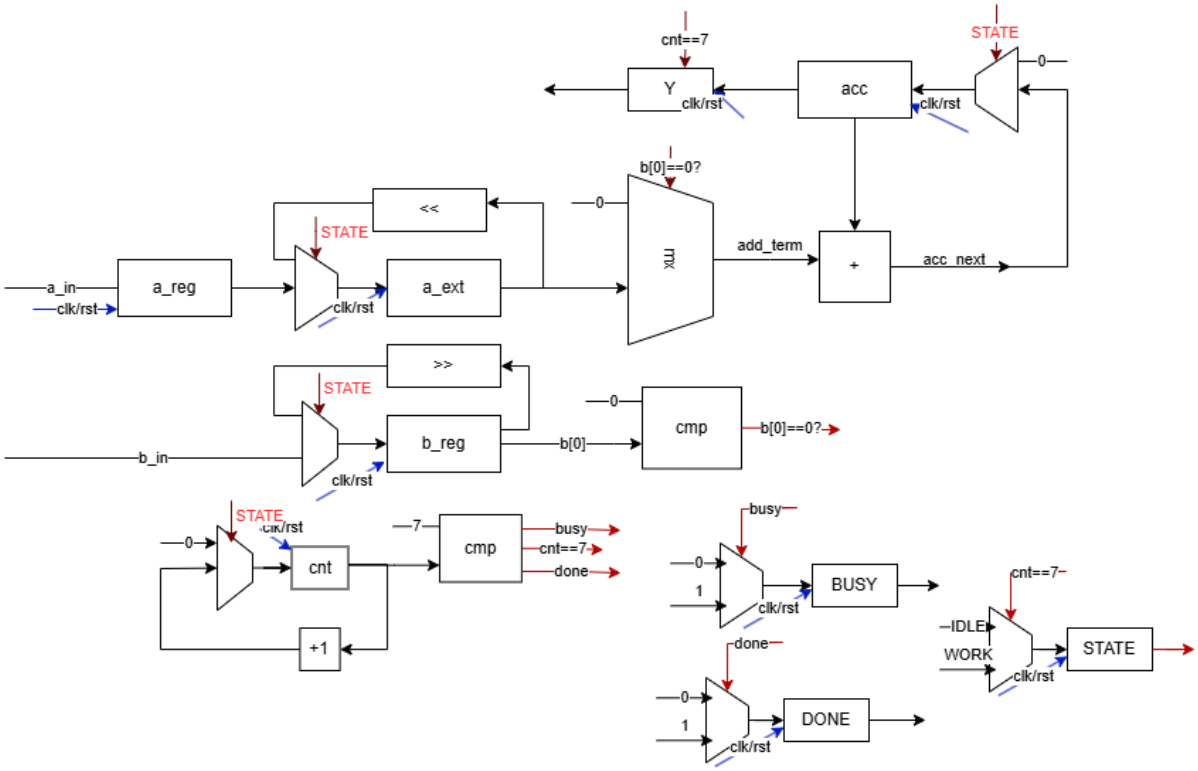
Отчет о проделанной работе

Область допустимых значений

8-битные входные значения – [0; 255]

24-битное выходное значение [0; 16646400]

Умножитель 16x8



Описание работы.

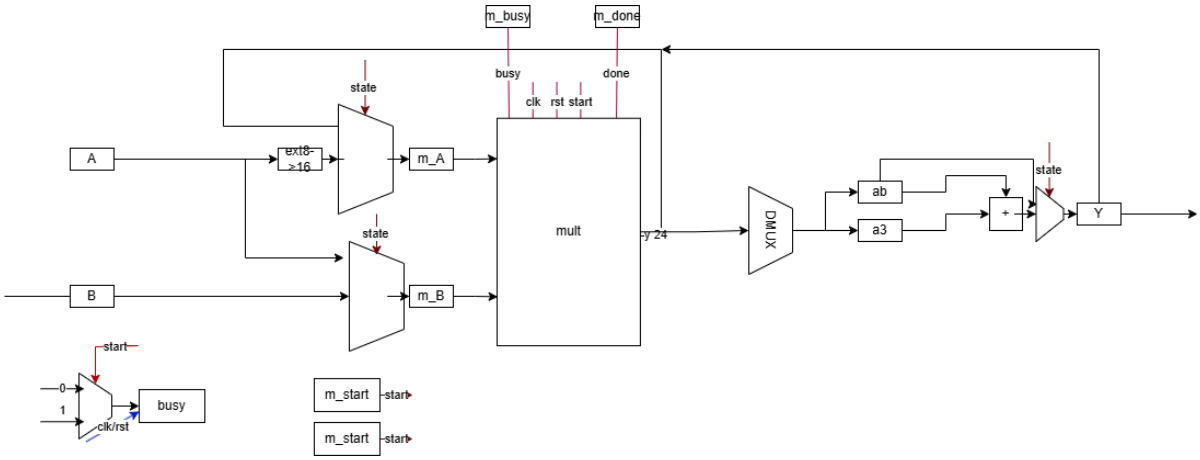
Подается стартовый сигнал, переход в состояние WORK. Защелкиваем значения на входах в a_ext (расширяя битность до 24) и b_reg, обнуляем регистры аккумулятора.
В состоянии WORK на каждом такте смотрим на 0 бит множителя b_reg, если он не равен нулю, то прибавляем к acc значение a_ext. a_ext сдвигаем влево на 1 бит, b_reg вправо на 1 бит. Увеличиваем cnt на 1. Если cnt==7, то умножение завершено, в регистр Y попадает результат умножения, BUSY=0, DONE=1, STATE=IDLE.

```
module mult16x8 #(
  parameter WA = 16, WB = 8
)(
  input  wire          clk,
  input  wire          rst,
```

```
 input  wire             start,
 input  wire [WA-1:0]      A,
 input  wire [WB-1:0]      B,
 output reg              busy,
 output reg              done,
 output reg  [WA+WB-1:0]    Y
);
 localparam WOUT = WA + WB;

 reg [WOUT-1:0] a_ext;
 reg [WB-1:0]   b_reg;
 reg [WOUT-1:0] acc;
 reg [7:0]     cnt;

 wire [WOUT-1:0] add_term = b_reg[0] ? a_ext : {WOUT{1'b0}};
 wire [WOUT-1:0] acc_next = acc + add_term;

 localparam IDLE=1'b0, WORK=1'b1;
 reg state;

 always @(posedge clk) begin
  if (rst) begin
   state<=IDLE; busy<=0; done<=0; Y<=0;
   a_ext<=0; b_reg<=0; acc<=0; cnt<=0;
  end else begin
   done <= 1'b0;
   case (state)
    IDLE: begin
     if (start) begin
      busy  <= 1'b1;
      a_ext <= {{(WOUT-WA){1'b0}}, A};
      b_reg <= B;
      acc   <= {WOUT{1'b0}};
      cnt   <= 0;
      state <= WORK;
     end
    end
    WORK: begin
     acc   <= acc_next;
     a_ext <= a_ext << 1;
     b_reg <= b_reg >> 1;
     cnt   <= cnt + 1;
     if (cnt == (WB-1)) begin
      Y    <= acc_next;
      busy <= 1'b0;
      done <= 1'b1;
      state<= IDLE;
```

```
      end
     end
    endcase
  end
 end
endmodule
```

Функция $ab+a^3$



Описание работы

```
module func (
 input  wire      clk,
 input  wire      rst,
 input  wire      start,
 input  wire [7:0]  a,
 input  wire [7:0]  b,
 output reg       busy,
 output reg       valid,
 output reg  [23:0] y
);
 reg       m_start;
 wire      m_busy, m_done;
 reg  [15:0] m_A;
 reg  [7:0]  m_B;
 wire [23:0] m_Y;

 mult16x8 mul (
  .clk(clk), .rst(rst),
  .start(m_start),
  .A(m_A), .B(m_B),
  .busy(m_busy), .done(m_done),
  .Y(m_Y)
 );

 reg [7:0]   a_r, b_r;
```

```
 reg [23:0]  ab24, a3_24;
 reg [15:0]  a2_16;

 localparam S_IDLE=3'd0, S_AB=3'd1, S_A2=3'd2, S_A3=3'd3, S_ADD=3'd4,
S_DONE=3'd5;
 reg [2:0] st;

 always @(posedge clk) begin
  if (rst) begin
   st<=S_IDLE; busy<=0; valid<=0; y<=0;
   m_start<=0; m_A<=0; m_B<=0;
   a_r<=0; b_r<=0; ab24<=0; a2_16<=0; a3_24<=0;
  end else begin
   m_start <= 1'b0;
   valid   <= 1'b0;
   case (st)
    S_IDLE: begin
     busy <= 1'b0;
     if (start) begin
      busy <= 1'b1;
      a_r  <= a; b_r <= b;

      m_A    <= {8'b0, a};
      m_B    <= b;
      m_start<= 1'b1;
      st     <= S_AB;
     end
    end
    S_AB: if (m_done) begin
     ab24 <= m_Y;

     m_A    <= {8'b0, a_r};
     m_B    <= a_r;
     m_start<= 1'b1;
     st     <= S_A2;
    end
    S_A2: if (m_done) begin
     a2_16 <= m_Y[15:0];

     m_A    <= {8'b0, m_Y[15:0]};
     m_B    <= a_r;
     m_start<= 1'b1;
     st     <= S_A3;
    end
    S_A3: if (m_done) begin
     a3_24 <= m_Y;
     st    <= S_ADD;
```

```
      end
      S_ADD: begin
       y  <= ab24 + a3_24;
       st  <= S_DONE;
      end
      S_DONE: begin
       valid <= 1'b1;
       busy  <= 1'b0;
       st   <= S_IDLE;
      end
    endcase
  end
 end
endmodule
```

Тестовое окружение

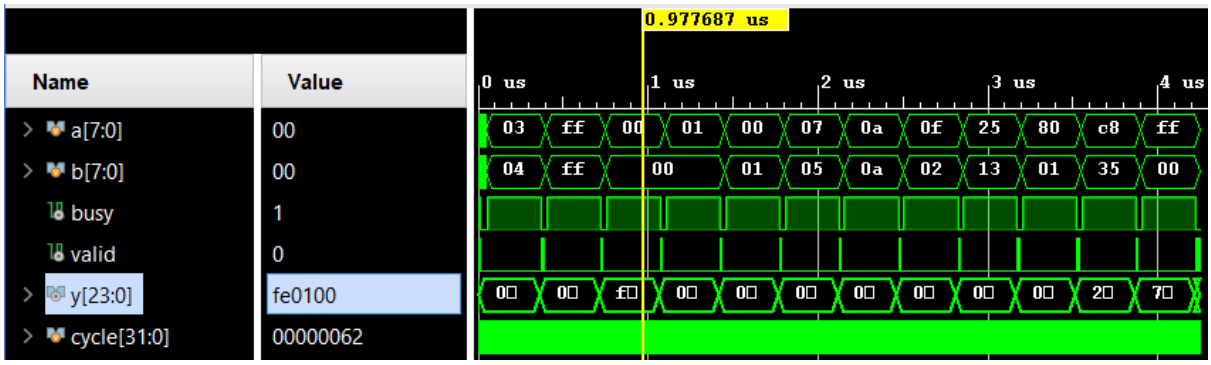
```
`timescale 1ns/1ps

module func_tb;
 reg clk = 0;
 reg rst = 1;

 reg      start = 0;
 reg [7:0]  a = 0, b = 0;

 wire      busy;
 wire      valid;
 wire [23:0] y;

 func dut (
  .clk(clk), .rst(rst),
  .start(start),
  .a(a), .b(b),
  .busy(busy), .valid(valid), .y(y)
 );

 always #5 clk = ~clk;

 integer cycle = 0;
 always @(posedge clk) cycle <= cycle + 1;

 task run_case(input [7:0] ta, input [7:0] tb, input [23:0] expected, input [255:0]
name);
    integer c_start, c_done, lat;
 begin
    @(negedge clk);
    a <= ta; b <= tb;
```

```
    start <= 1'b1;
    c_start = cycle;
    @(negedge clk);
    start <= 1'b0;

    @(posedge valid);
    c_done = cycle;
    lat = c_done - c_start;

    if (y !== expected) begin
      $display("[FAIL] %0s: a=%0d b=%0d  got=%0d exp=%0d", name, ta, tb, y,
expected);
      $fatal(1);
    end

    $display("[OK] ? %0s: y=%0d  LAT = %0d cycles  (~%0d ns)",
          name, y, lat, lat*10);

    repeat(2) @(posedge clk);
   end
   endtask


 initial begin
 $dumpfile("func_tb.vcd");
 $dumpvars(0, func_tb);

 repeat(5) @(posedge clk);
 rst <= 1'b0;

 run_case(8'd3,   8'd4,   24'd39,      "SMALL");
 run_case(8'd255, 8'd255, 24'd16646400,  "MAX");

 run_case(8'd0,   8'd0,   24'd0,       "T0");
 run_case(8'd1,   8'd0,   24'd1,       "T1");       // 1*0 + 1^3 = 1
 run_case(8'd0,   8'd1,   24'd0,       "T2");       // 0*1 + 0^3 = 0
 run_case(8'd7,   8'd5,   24'd378,     "T3");       // 7*5 + 343 = 378
 run_case(8'd10,  8'd10,  24'd1100,    "T4");       // 100 + 1000
 run_case(8'd15,  8'd2,   24'd3405,    "T5");       // 30 + 3375
 run_case(8'd37,  8'd19,  24'd51356,   "T6");       // 703 + 50653
 run_case(8'd128, 8'd1,   24'd2097280,  "T7");      // 128 + 2,097,152
 run_case(8'd200, 8'd53,  24'd8010600,  "T8");      // 10,600 + 8,000,000
 run_case(8'd255, 8'd0,   24'd16581375,  "T9");     // 0 + 255^3

 $finish;

 end
```

```
endmodule
```

Результат тестирования

```
run -all
[OK] ? SMALL: y=39  LAT = 33 cycles  (~330 ns)
[OK] ? MAX: y=16646400  LAT = 33 cycles  (~330 ns)
[OK] ? T0: y=0  LAT = 33 cycles  (~330 ns)
[OK] ? T1: y=1  LAT = 33 cycles  (~330 ns)
[OK] ? T2: y=0  LAT = 33 cycles  (~330 ns)
[OK] ? T3: y=378  LAT = 33 cycles  (~330 ns)
[OK] ? T4: y=1100  LAT = 33 cycles  (~330 ns)
[OK] ? T5: y=3405  LAT = 33 cycles  (~330 ns)
[OK] ? T6: y=51356  LAT = 33 cycles  (~330 ns)
[OK] ? T7: y=2097280  LAT = 33 cycles  (~330 ns)
[OK] ? T8: y=8010600  LAT = 33 cycles  (~330 ns)
[OK] ? T9: y=16581375  LAT = 33 cycles  (~330 ns)
```



Выводы

В ходе работы был спроектирован модуль умножителя, один его вход был расширен до 16 разрядов, т. к. для вычисления a^3 необходимо сохранить промежуточный результат вычисления квадрата. Этот умножитель был использован для разработки схему вычисления функции $a*b+a^3$.

По результатам тестирования схема показала корректные вычисления на разных тестовых значениях. Вычисление функции всегда занимает 33 такта (3 умножения по 10 тактов, 1 такт сложение и 2 такта на выход в DONE/valid), так как данный последовательный умножитель делает 8 итераций независимо от данных.