### Санкт-Петербургский государственный политехнический университет Институт компьютерных наук и технологий

**Отчёт по лабораторной работе:**

**“Вычисление определителя матрицы размерностью 3 на 3”**

**По дисциплине:**

**«Микроэлектроника, схемотехника и проектирование устройств вычислительной техники»**

Выполнили студенты гр. 3530904/90002 : Ли Ицзя

Мэн Цзянин

Го Синлун

Научный руководитель Амосов В.В.

### Санкт-Петербург 2020

# 1.Техническое задание

### сосчитать определитель матрицы размера 3 на 3 для модели процессора DP32.

# 

# Выполнение работы

## 

## 1.Расчёт определителя матрицы

### Определитель матрицы размера 3 на 3:

### http://mathprofi.ru/f/kak_vychislit_opredelitel_clip_image022.gif

## 

## 2.Создание проекта в Active-VHDL

### На основе кода процессора, представленного в 22 главе, создан проект DP32 в среде Active-VHDL. Он состоит из следующих файлов:

### dp32\_behav.vhd — поведенческая модель процессора;

### memory.vhd — модель памяти;

### clk\_gen.vhd — модель генератора тактовой частоты;

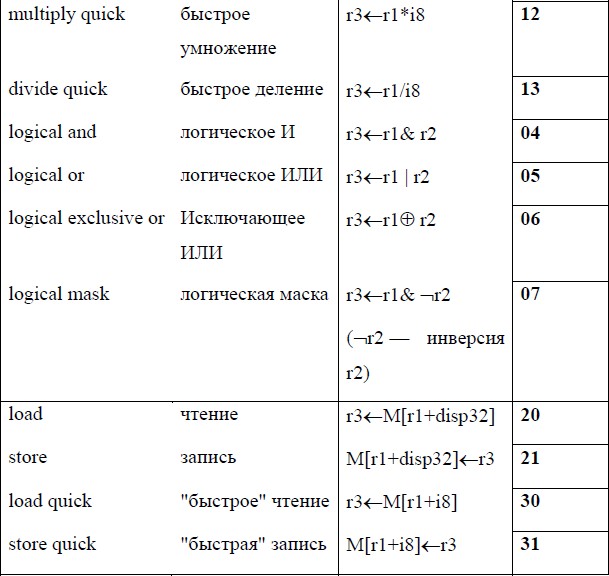
### dp32\_test.vhd — модель теста;

### dp32\_behaviour\_test — конфигурация для VHDL-модели теста поведенческой модели DP32;

### alu.vhd — модель АЛУ.

## Написание кода программы

### Код программы писался на основании операций, представленных в следующей таблице



### Текст программы на машинном коде:

### строки;

## Описание алгоритма программы

### Обнуляем r0;

### – (3) Вычисляем произведение 𝑎1𝑏2𝑐3 в r2, записываем результат в r1;

### (4) – (7) Вычисляем произведение 𝑎3𝑏1𝑐2 в r2, прибавляем результат к r1;

### (8) – (11) Вычисляем произведение 𝑎2𝑏3𝑐1 в r2, прибавляем результат к r1;

### (12) – (15) Вычисляем произведение 𝑎3𝑏2𝑐1 в r2, вычитаем результат из r1;

### (16) – (19) Вычисляем произведение 𝑎2𝑏1𝑐3 в r2, вычитаем результат из r1;

### (20) – (23) Вычисляем произведение 𝑎1𝑏3𝑐2 в r2, вычитает результат из r1;

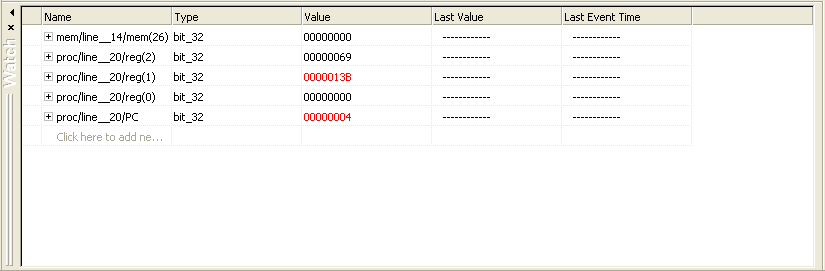
### Записываем результат вычислений из r1 в ячейку памяти с адресом из следующей

### Смещение равно 26 (в десятичной записи);

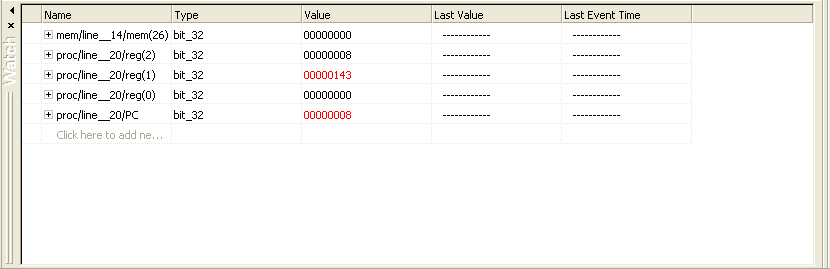


## Результат работы программы

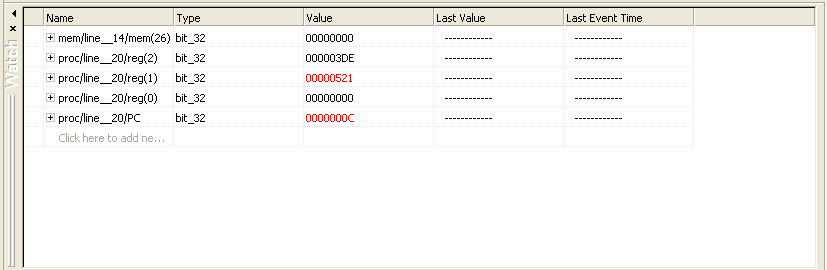
### скриншоты, иллюстрирующие работу программы:



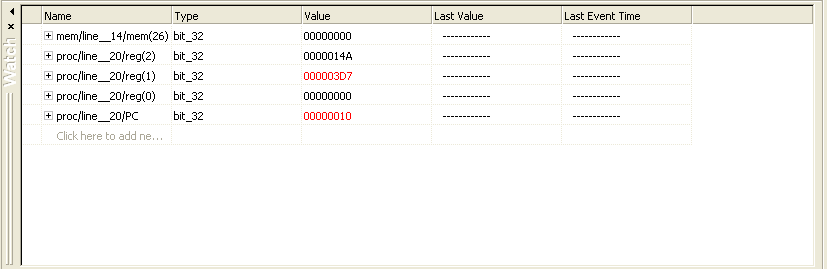
**Рис. 1. Запись произведения a1b2c3 в r1**



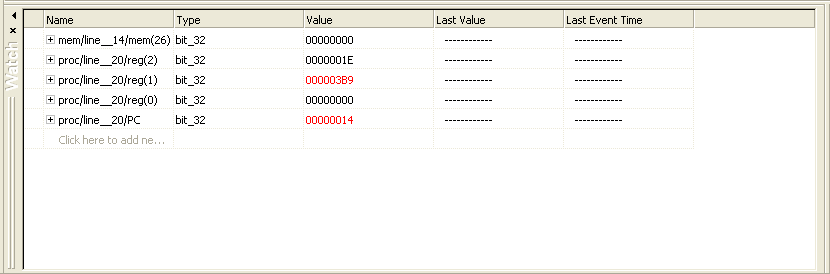
**Рис. 2. Прибавление произведения a3b1c2 к r1**



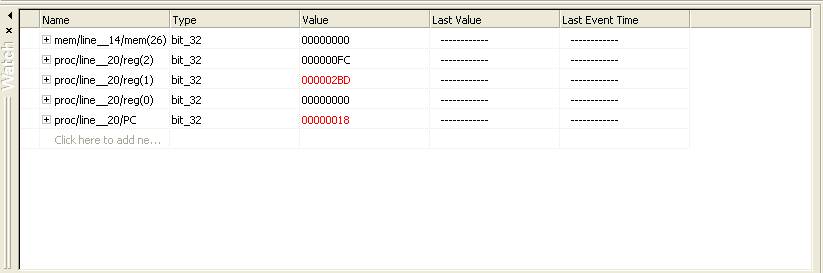
**Рис. 3. Прибавление произведенияa2b3c1 к r1**



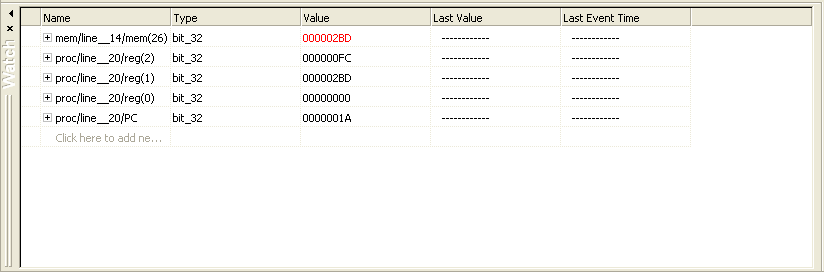
**Рис. 4. Вычитание произведения a3b2c1 из r1**



**Рис. 5. Вычитание произведения a2b1c3 из r1**



**Рис. 6. Вычитание произведения a1b3c2 из r1**



**Рис. 7. Запись значения из r1 в ячейку памяти по адресу 26**

## Ручной расчёт результата

### Ответ, полученный в результате работы программы, равен 2BD(16) или 701(10). Произведём ручной расчёт для проверки результата:

### Det = 7 \* 15 \* 3 + 10 \* 9 \* 11 + 2 \* 1 \* 4 – 2 \* 15 \* 11 – 10 \* 1 \* 3 – 7 \* 9 \* 4 = 315 + 990 + 8 –

### 330 – 30 – 252 = 701

### Приведённые расчёты показывают соответствие результата и работы программы условию задания.

## 

# Реализация задачи на языке VHDL

## Содержание проекта

### В ходе работы было создано два файла, регулирующие работу программы, и один для тестирования программы:

### header.vhd – файл, в котором находится пакет, необходимый для работы поведенческой модели.

### Solution.vhd – файл с реализацией поведенческой модели, где get\_determinant – функция, возвращающая значение определителя матрицы, функции vec\_to\_int и int\_to\_vec осуществляют перевод в двоичную и десятичную системы счисления.

### test\_solution.vhd – файл, содержащий модуль тестирования работы программы. Код сгенерирован в программе Active-VHDL автоматически.

## Листинг header.vhd

library IEEE;

use IEEE.std\_logic\_1164.all; package header is

-- Type of array for numeric progression

TYPE matrix\_vector is array (0 to 2, 0 to 2) of

std\_logic\_vector (31 downto 0);

TYPE matrix\_integer is array (0 to 2, 0 to 2) of integer;

-- Calculate determinant

procedure get\_determinant (a : in matrix\_vector;

result : out

std\_logic\_vector (31 downto 0));

-- Binary translation

function vec\_to\_int (vec : in std\_logic\_vector (31 downto 0))

return integer;

-- Decimal translation

procedure int\_to\_vec (num : in integer;

vec : out std\_logic\_vector (31 downto 0));

end header;

package body header is

-- Calculate determinant

procedure get\_determinant (a : in matrix\_vector;

result : out

std\_logic\_vector (31 downto 0)) is

variable i, j, tmp\_result : integer; variable tmp : matrix\_integer; begin

for i in 0 to 2 loop

for j in 0 to 2 loop

tmp(i, j) := vec\_to\_int(a(i, j)); end loop;

end loop;

tmp\_result := tmp(0, 0) \* tmp(1, 1) \* tmp(2, 2); tmp\_result := tmp\_result + tmp(0, 1) \* tmp(1, 2) \*

tmp(2, 0);

tmp(2, 1);

tmp\_result := tmp\_result + tmp(0, 2) \* tmp(1, 0) \*

tmp(2, 0);

tmp(2, 2);

tmp(2, 1);

tmp\_result := tmp\_result - tmp(0, 2) \* tmp(1, 1) \* tmp\_result := tmp\_result - tmp(0, 1) \* tmp(1, 0) \* tmp\_result := tmp\_result - tmp(0, 0) \* tmp(1, 2) \*

int\_to\_vec(tmp\_result, result); end get\_determinant;

-- Binary translation

function vec\_to\_int (vec : in std\_logic\_vector (31 downto 0))

return integer is variable result : integer := 0;

variable tmp : integer range 0 to 1 := 0; variable word : std\_logic\_vector (31 downto 0); begin

if vec(vec'left) = '1' then

-- Number is negative word := not vec;

else

-- Number is positive word := vec;

end if;

for ind in vec'range loop if vec(ind) = '0' then

tmp := 0;

else

tmp := 1;

end if;

result := result \* 2 + tmp; end loop;

if vec(vec'left) = '1' then

-- Number is negative return (-result)-1;

else

-- Number is positive return result;

end if;

end;

-- Decimal translation

procedure int\_to\_vec (num : in integer;

vec : inout std\_logic\_vector (31 downto 0))

is

variable temp : integer := 0; variable ind : integer;

begin

if num = 0 then

vec := "00000000000000000000000000000000";

else

if num < 0 then

-- Number is negative temp := -(num + 1);

else

-- Number is positive temp := num;

end if;

for ind in vec'reverse\_range loop if (temp rem 2 = 0) then

vec(ind) := '0';

else

vec(ind) := '1';

end; end header;

end if;

temp := temp / 2; end loop;

if num < 0 then

-- Number in negative vec := not vec; vec(vec'left) := '1';

end if; end if;

## Листинг solution.vhd

use work.header.all;

library IEEE;

use IEEE.std\_logic\_1164.all;

entity solution is

port(a : in matrix\_vector;

signal result : out std\_logic\_vector (31 downto 0)); end solution;

architecture behavior of solution is begin

process(a)

variable tmp\_result : std\_logic\_vector (31 downto 0);

begin

get\_determinant(a, tmp\_result); result <= tmp\_result;

end process; end behavior;

## Листинг solution\_tb.vhd

--\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

--\* This file is automatically generated test bench template \*

--\* By ACTIVE-VHDL <TBgen v1.10>. Copyright (C) ALDEC Inc. \*

--\* \*

--\* This file was generated on: 0:43, 31.05.2018 \*

--\* Tested entity name: solution \*

--\* File name contains tested entity: .\src\Task\solution.vhd \*

--\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

library ieee;

use work.header.all;

use ieee.std\_logic\_1164.all;

-- Add your library and packages declaration here ... entity solution\_tb is

end solution\_tb;

architecture TB\_ARCHITECTURE of solution\_tb is

-- Component declaration of the tested unit component solution

port(

a : in matrix\_vector;

result : out std\_logic\_vector(31 downto 0) ); end component;

-- Stimulus signals - signals mapped to the input and inout ports of tested entity

signal a : matrix\_vector;

-- Observed signals - signals mapped to the output ports of tested entity

signal result : std\_logic\_vector(31 downto 0);

-- Add your code here ...

begin

-- Unit Under Test port map UUT : solution

port map

(a => a,

result => result );

-- Add your stimulus here ...

end TB\_ARCHITECTURE;

configuration TESTBENCH\_FOR\_solution of solution\_tb is for TB\_ARCHITECTURE

for UUT : solution

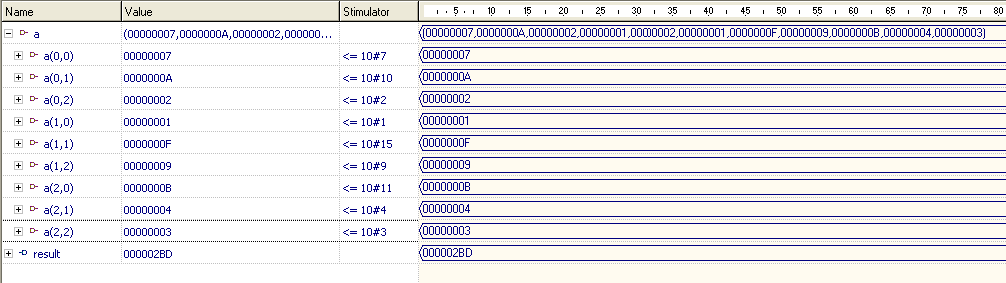
use entity work.solution(behavior); end for;

end for;

end TESTBENCH\_FOR\_solution;

## Результат тестирования

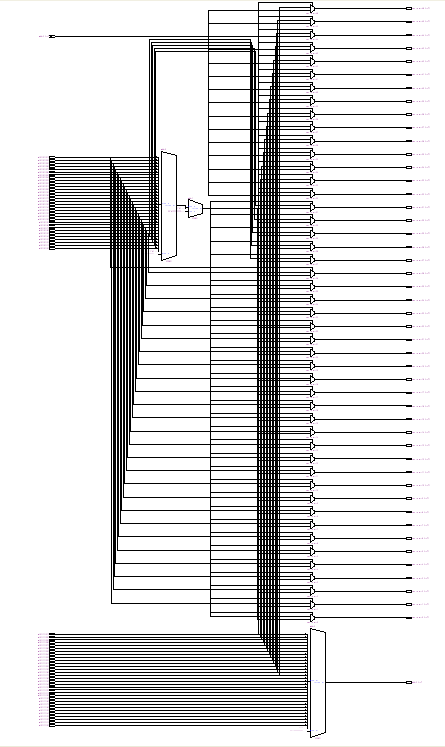
### В ходе тестирования был получен результат, соответствующий ожиданиям:



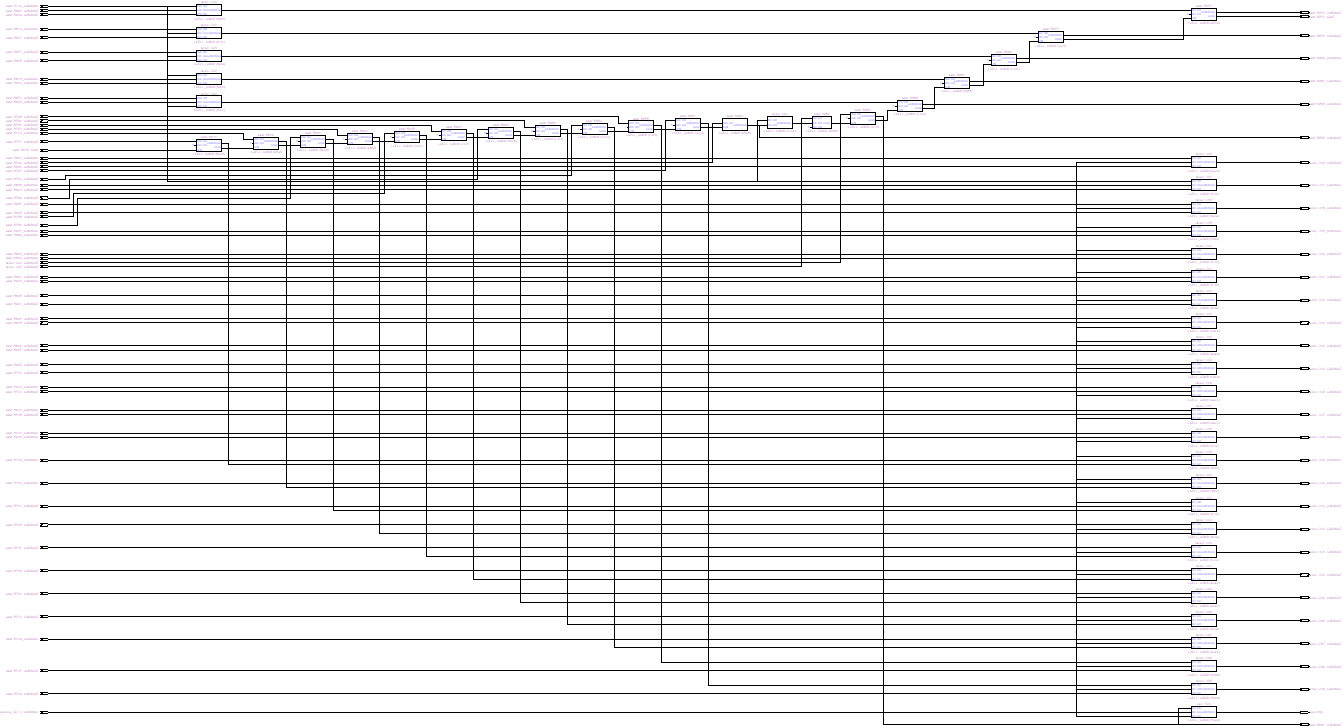
**Рис. 8. Результат тестирования программы, написанной на VHDL**

## Схемы реализации

### С помощью инструментов Quartus II 5.0 были получены RTL-схема и техническая схема.



**Рис. 9. RTL-схема**



**Рис. 10. Tехническая схема**

### 