Санкт-Петербургский государственный политехнический университет Институт компьютерных наук и технологий
~~“Высшая школа программной инженерии”~~

Отчёт по лабораторной работе:
“Вычисление определителя матрицы размерностью 3 на 3”

По дисциплине:
«Микроэлектроника, схемотехника и проектирование устройств вычислительной техники»

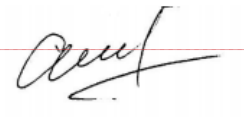
Выполнили студенты гр. 3530904/90002 ~~13534/7~~:

Ли Ицзя
Мэн Цзянин
Го Синлун

~~Андреев А.С.~~

~~Каширин А.И.~~

Научный руководитель
Амосов В.В.



样式定义: 标题 2: 字体: 倾斜
设置了格式: 字体: (默认) Times New Roman
设置了格式: 字体: (默认) Times New Roman
设置了格式: 字体: (默认) Times New Roman
带格式的: 右侧: 2 厘米, 段落间距段前: 0 磅
带格式的: 正文文本, 左
设置了格式: 字体: (默认) Times New Roman

Санкт-Петербург 2020

1.Техническое задание

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

$$detA = \begin{vmatrix} 7 & 10 & 2 \\ 1 & 15 & 9 \\ 11 & 4 & 3 \end{vmatrix}$$

Выполнение работы

1.Расчёт определителя матрицы

Определитель матрицы размера 3 на 3:

$$\begin{vmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \end{vmatrix} = a_1b_2c_3 + a_3b_1c_2 + a_2b_3c_1 - a_3b_2c_1 - a_1b_3c_2 - a_2b_1c_3$$

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:

1. dp32_behav.vhd — поведенческая модель процессора;
2. memory.vhd — модель памяти;
3. clk_gen.vhd — модель генератора тактовой частоты;
4. dp32_test.vhd — модель теста;
5. dp32_behaviour_test — конфигурация для VHDL-модели теста поведенческой модели DP32;
6. alu.vhd — модель АЛУ.

3. Написание кода программы

设置了格式: 字体: (默认) Times New Roman

设置了格式: 字体: 14 磅

设置了格式: 字体: 14 磅

设置了格式: 字体: 14 磅

设置了格式: 字体: 14 磅

设置了格式: 字体: 14 磅

设置了格式: 字体: 14 磅

设置了格式: 字体: 14 磅

设置了格式: 字体: 14 磅

设置了格式: 字体: 4 磅

设置了格式: 字体: 14 磅

带格式的: 左

带格式的: 缩进: 左侧: 1.45 厘米, 悬挂缩进: 0.42 厘米

Код программы писался на основании операций, представленных в следующей таблице

Название		Действие	Код операции
add	сложение	r3←r1+r2	**00**
subtract	вычитание	r3←r1-r2	**01**
multiply	умножение	r3←r1*r2	**02**
divide	деление (нацело)	r3←r1/r2	**03**
add quick	быстрое сложение	r3←r1+i8	**10**
subtract quick	быстрое вычитание	r3←r1-i8	**11**

multiply quick	быстрое умножение	r3←r1*i8	**12**
divide quick	быстрое деление	r3←r1/i8	**13**
logical and	логическое И	r3←r1& r2	**04**
logical or	логическое ИЛИ	r3←r1 | r2	**05**
logical exclusive or	Исключающее ИЛИ	r3←r1⊕ r2	**06**
logical mask	логическая маска	r3←r1& ¬r2 (¬r2 — инверсия r2)	**07**
load	чтение	r3←M[r1+disp32]	**20**
store	запись	M[r1+disp32]←r3	**21**
load quick	"быстрое" чтение	r3←M[r1+i8]	**30**
store quick	"быстрая" запись	M[r1+i8]←r3	**31**

Текст программы на машинном коде:

```
variable mem :memory_array:=
    (X"0700_0000", --  0. r0 = 0
     X"1002_0007", --  1. r2 = r0 + 7
     X"1202_020F", --  2. r2 = r2 * 15
     X"1201_0203", --  3. r1 = r2 * 3
     X"1002_0001", --  4. r2 = r0 + 1
     X"1202_0204", --  5. r2 = r2 * 4
     X"1202_0202", --  6. r2 = r2 * 2
     X"0001_0102", --  7. r1 = r1 + r2
     X"1002_000B", --  8. r2 = r0 + 11
     X"1202_020A", --  9. r2 = r2 * 10
     X"1202_0209", -- 10. r2 = r2 * 9
     X"0001_0102", -- 11. r1 = r1 + r2
     X"1002_000B", -- 12. r2 = r0 + 10
     X"1202_020F", -- 13. r2 = r2 * 15
     X"1202_0202", -- 14. r2 = r2 * 2
     X"0101_0102", -- 15. r1 = r1 - r2
     X"1002_0001", -- 16. r2 = r2 + 1
     X"1202_020A", -- 17. r2 = r2 * 10
     X"1202_0203", -- 18. r2 = r2 * 3
     X"0101_0102", -- 19. r1 = r1 - r2
     X"1002_0007", -- 20. r2 = r0 + 7
     X"1202_0209", -- 21. r2 = r2 * 9
     X"1202_0204", -- 22. r2 = r2 * 4
     X"0101_0102", -- 23. r1 = r1 - r2
     X"2101_0000", -- 24. Write r1 to disp32
     X"0000_001A", -- 25. disp32 = 26
     others =>X"0000_0000");
```

4. Описание алгоритма программы

(0) Обнуляем r0;

(1) – (3) Вычисляем произведение $a_1 b_2 c_3$ в r2, записываем результат в r1;

(4) – (7) Вычисляем произведение $a_3 b_1 c_2$ в r2, прибавляем результат к r1;

(8) – (11) Вычисляем произведение $a_2 b_3 c_1$ в r2, прибавляем результат к r1;

(12) – (15) Вычисляем произведение $a_3 b_2 c_1$ в r2, вычитаем результат из r1;

(16) – (19) Вычисляем произведение $a_2 b_1 c_3$ в r2, вычитаем результат из r1;

(20) – (23) Вычисляем произведение $a_1 b_3 c_2$ в r2, вычитает результат из r1;

(24) Записываем результат вычислений из r1 в ячейку памяти с адресом из следующей

строки;

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


Обнуляем r0
Считаем произведение элементов a1b2c3 в r2
Складываем r1 и r2
Считаем произведение a3b1c2 в r2
Складываем r1 и r2
Считаем произведение a2b3c1 в r2
Складываем r1 и r2
Считаем произведение a3b2c1 в r2
Вычитаем r2 из r1
Считаем произведение a2b1c3 в r2
Вычитаем r2 из r1
Считаем произведение a1b3c2в r2
Записываем результат вычислений из r1 в ячейку памяти с адресом 26

5. Результат работы программы

скриншоты, иллюстрирующие работу программы:

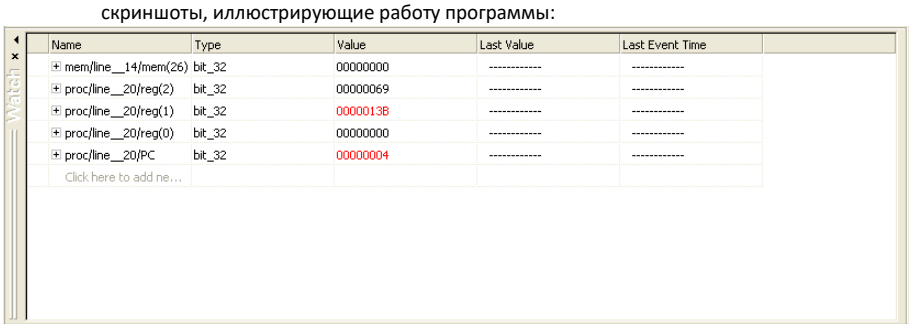


Рис. 1. Запись произведения $a_1b_2c_3$ в r1

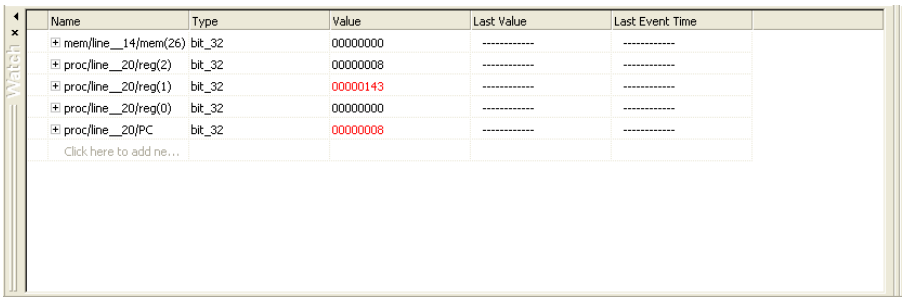


Рис. 2. Прибавление произведения $a_3b_1c_2$ к r1

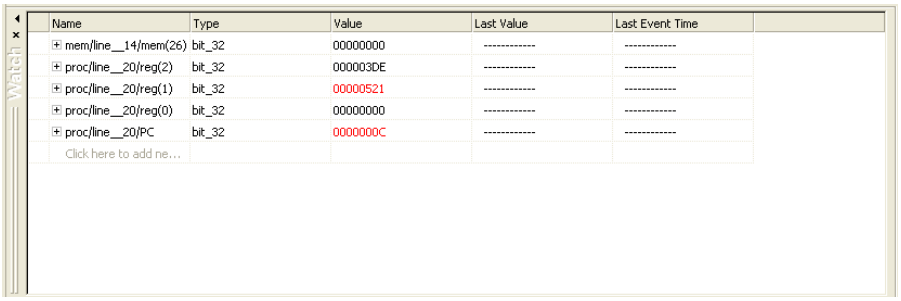


Рис. 3. Прибавление произведенияa2b3c1 к r1

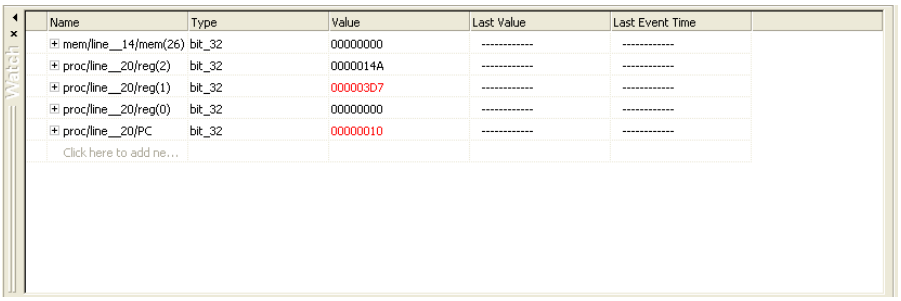


Рис. 4. Вычитание произведения $a_3b_2c_1$ из r1

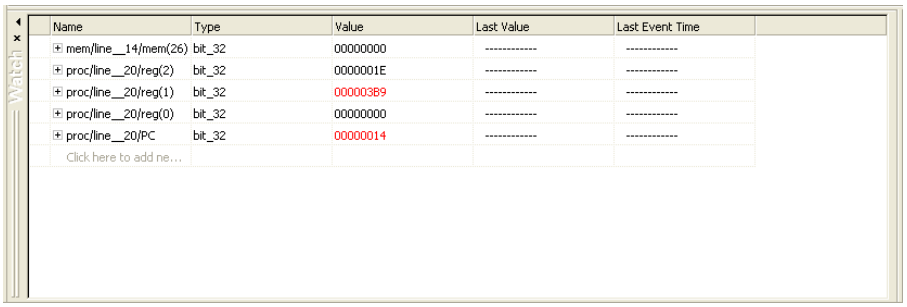


Рис. 5. Вычитание произведения $a_2b_1c_3$ из r1

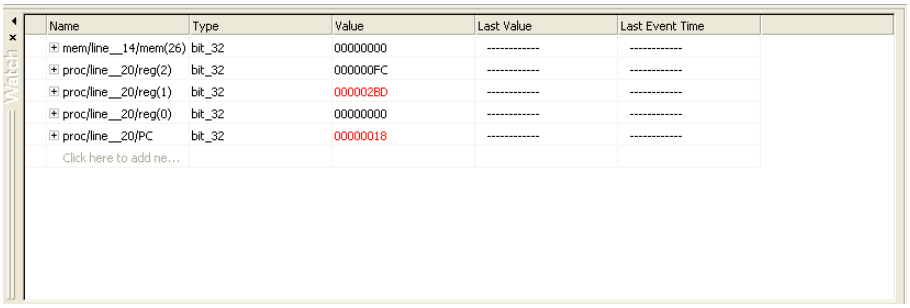


Рис. 6. Вычитание произведения $a_1b_3c_2$ из r1

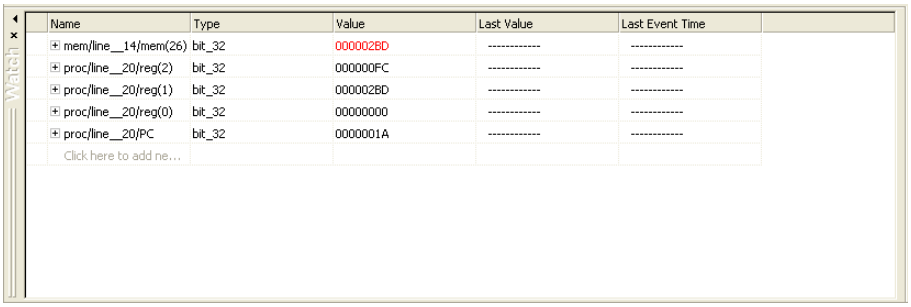


Рис. 7. Запись значения из r1 в ячейку памяти по адресу 26

6. 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

带格式的: 缩进: 左侧: 0 厘米

1. Содержание проекта

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

1. header.vhd – файл, в котором находится пакет, необходимый для работы поведенческой модели.
2. 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.
3. test_solution.vhd – файл, содержащий модуль тестирования работы программы. Код сгенерирован в программе Active-VHDL автоматически.

2. 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_result := tmp_result + tmp(0, 2) * tmp(1, 0) *
tmp(2, 1);
```

```
                        tmp_result := tmp_result - tmp(0, 2) * tmp(1, 1) *
tmp(2, 0);
                        tmp_result := tmp_result - tmp(0, 1) * tmp(1, 0) *
tmp(2, 2);
                        tmp_result := tmp_result - tmp(0, 0) * tmp(1, 2) *
tmp(2, 1);

                        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 if;

                      temp := temp / 2;
                end loop;

                if num < 0 then
                      -- Number in negative
                      vec := not vec;
                      vec(vec'left) := '1';
                end if;
          end if;
      end;
end header;
```

3. 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;
```

4. 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:

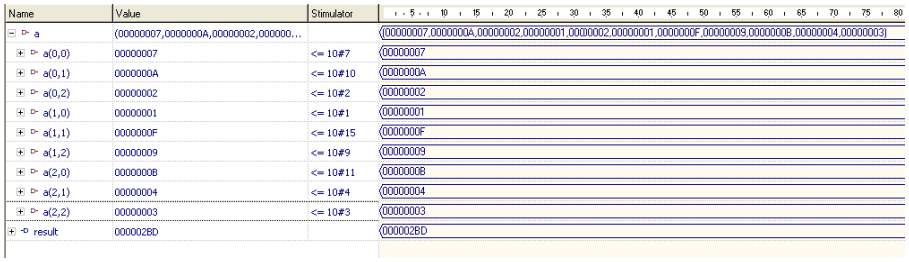


Рис. 8. Результат тестирования программы, написанной на VHDL

1. Схемы реализации

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

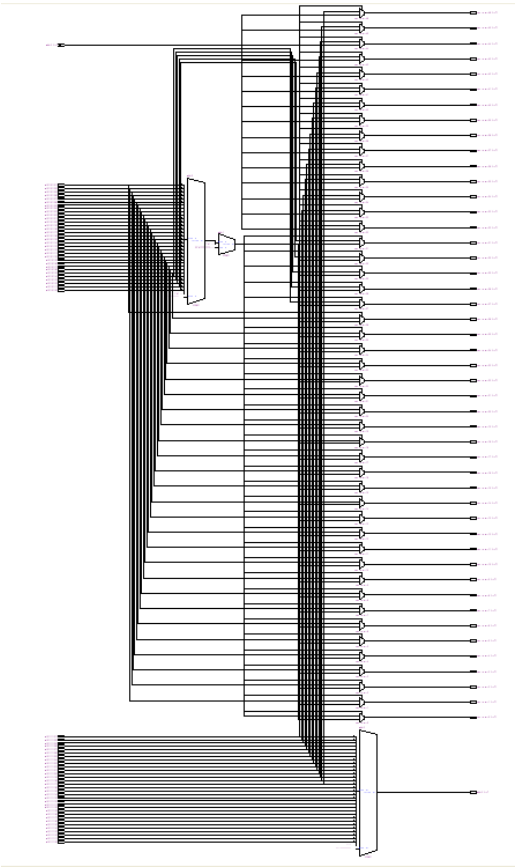


Рис. 9. RTL-схема

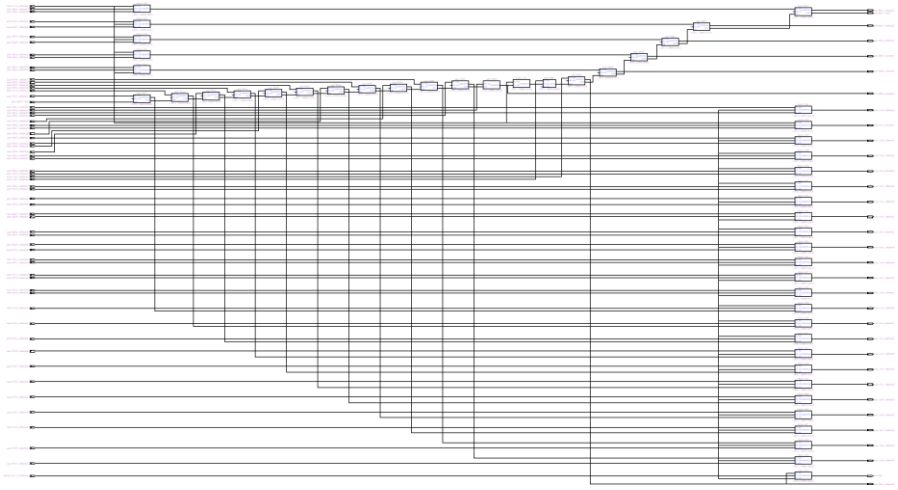


Рис. 10. Техническая схема