Министерство образования и науки РФ
Санкт-Петербургский политехнический университет Петра Великого
Институт компьютерных наук и технологий
Высшая школа программной инженерии

ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ
по дисциплине «Микроэлектроника, схемотехника
и проектирование устройств вычислительной техники»

Программная и аппаратная реализация алгоритма сложения матриц

Выполнили:
Студенты гр. _____

_____ _____

Проверил:

_____

Санкт-Петербург 2020 г.

Оглавление

1. Техническое задание

1) Разработать аппаратную реализацию сортировки пузырьком массива на языке VHDL
2) Произвести симуляцию VHDL кода
3) Создать RTL схему в среде Quartus II
4) Создать технологическую схему
5) Разработать программную реализацию сортировки вставками массива в кодах процессора DP32
6) Создать блок-схему алгоритма на кодах процессора DP32
7) Произвести ко-симуляцию в среде Active VHDL
8) Сделать ручной расчёт сортировки

2. Описание алгоритма

На вход алгоритма подаётся последовательность n чисел. Алгоритм состоит из повторяющихся проходов по сортируемому массиву. За каждый проход элементы последовательно сравниваются попарно и, если порядок в паре неверный, выполняется обмен элементов. Проходы по массиву повторяются n-1 раз или до тех пор, пока на очередном проходе не окажется, что обмены больше не нужны, что означает — массив отсортирован. При каждом проходе алгоритма по внутреннему циклу, очередной наибольший элемент массива ставится на своё место в конце массива рядом с предыдущим «наибольшим элементом», а наименьший элемент перемещается на одну позицию к началу массива.

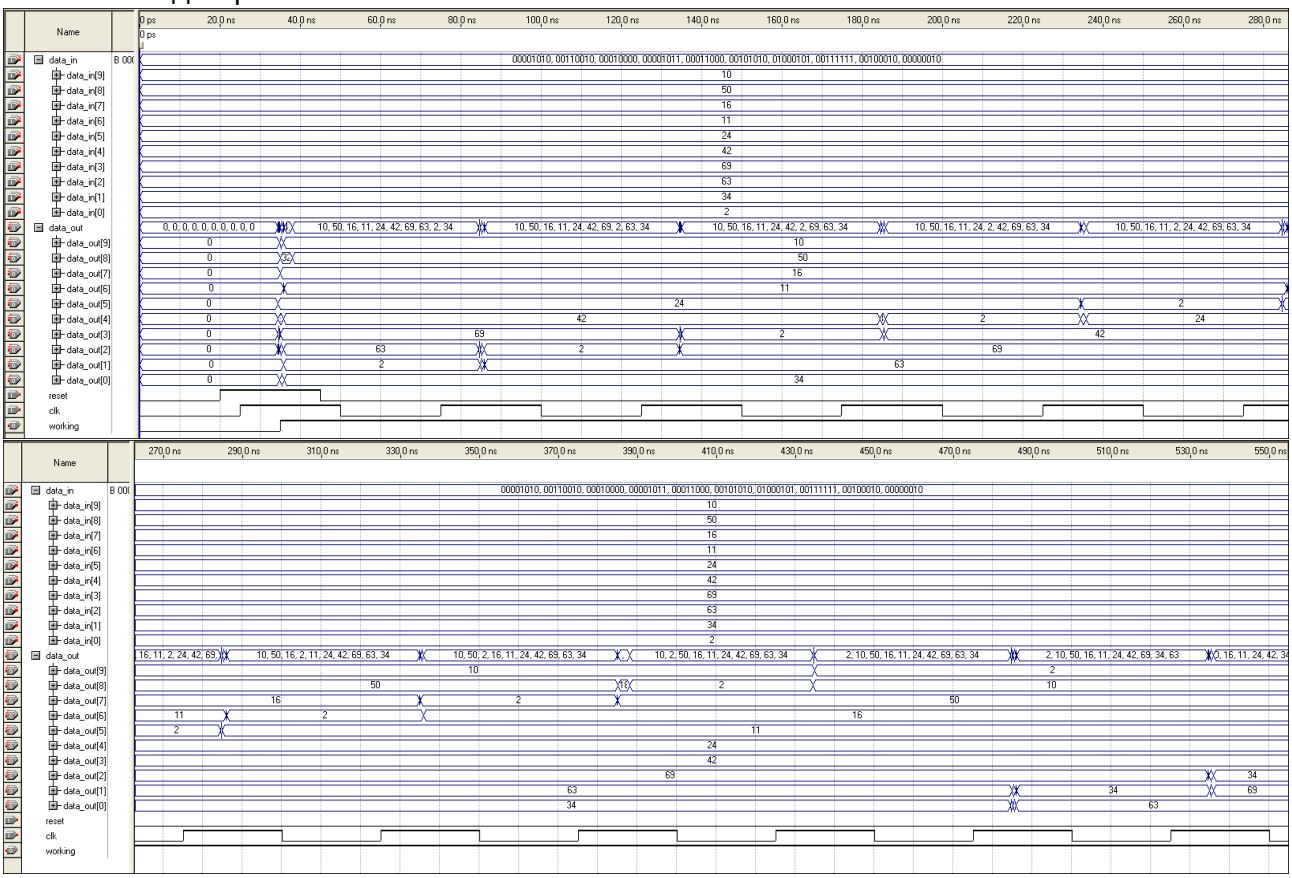
3. Аппаратная реализация

3.1 VHDL код

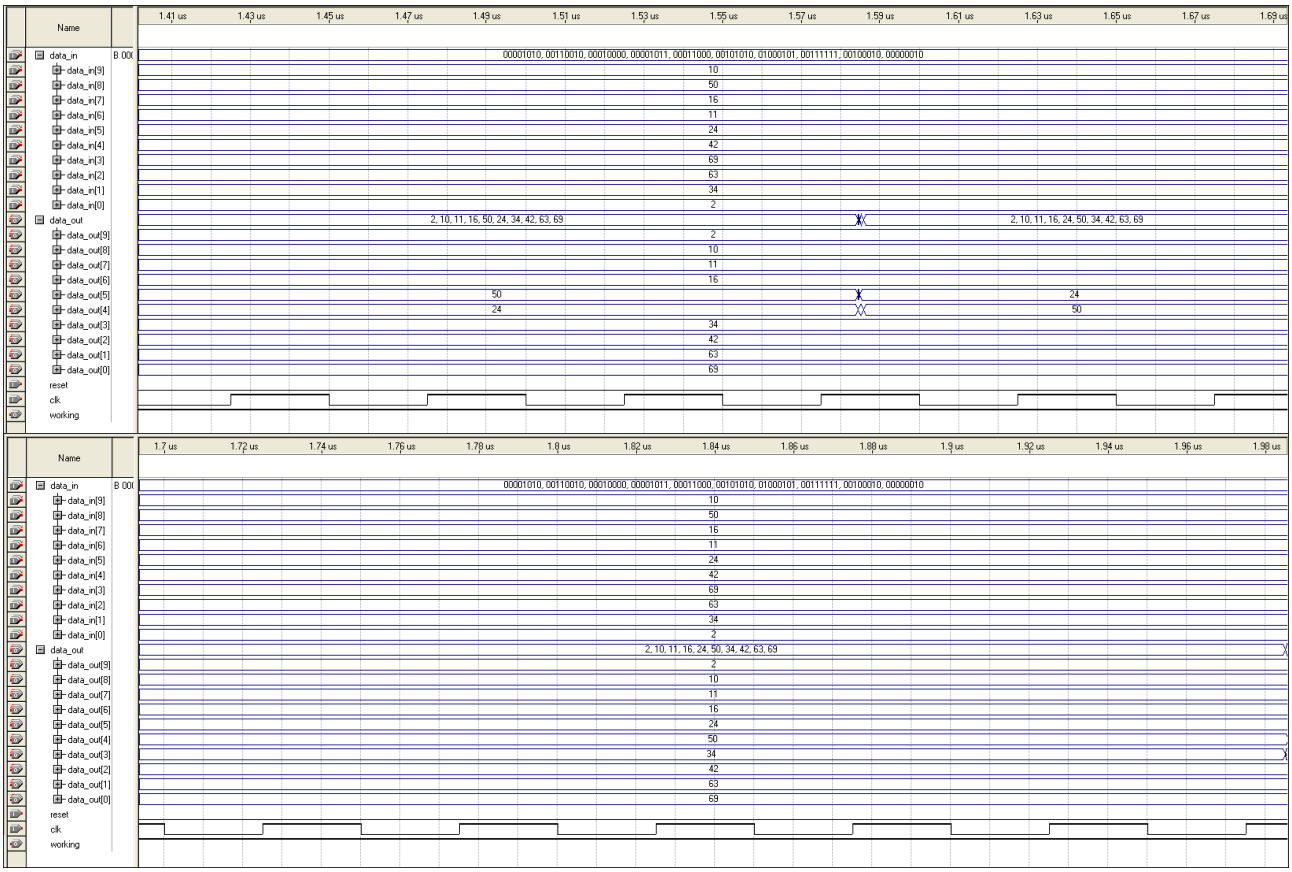
```
1  library ieee;
2  use ieee.std_logic_1164.all;
3
4  package sort_types is
5  type massType is
6      array (9 downto 0) of std_logic_vector (7 downto 0);
7  end sort_types;
8
9  library ieee;
10 use ieee.std_logic_1164.all;
11 use work.sort_types.all;
12 entity sort is
13     port (
14         clk: in std_logic;
15         reset: in std_logic;
16         working: out std_logic;
17         data_in: in massType;
18         data_out: out massType
19     );
20 end sort;
21 architecture beh of sort is
22     signal run: std_logic := '0';
23 begin
24     process (clk)
25         variable i, e: integer := 0;
26         variable arr: massType;
27     …….
28                                 data_out <= arr;
29                                 e := e + 1;
30                             end if;
31                             i := i + 1;
32     ……
33                                     data_out <= arr;
34                                     i := 1;
35                                 end if;
36                             end if;
37                     end if;
38     end process;
39 end beh;
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
```

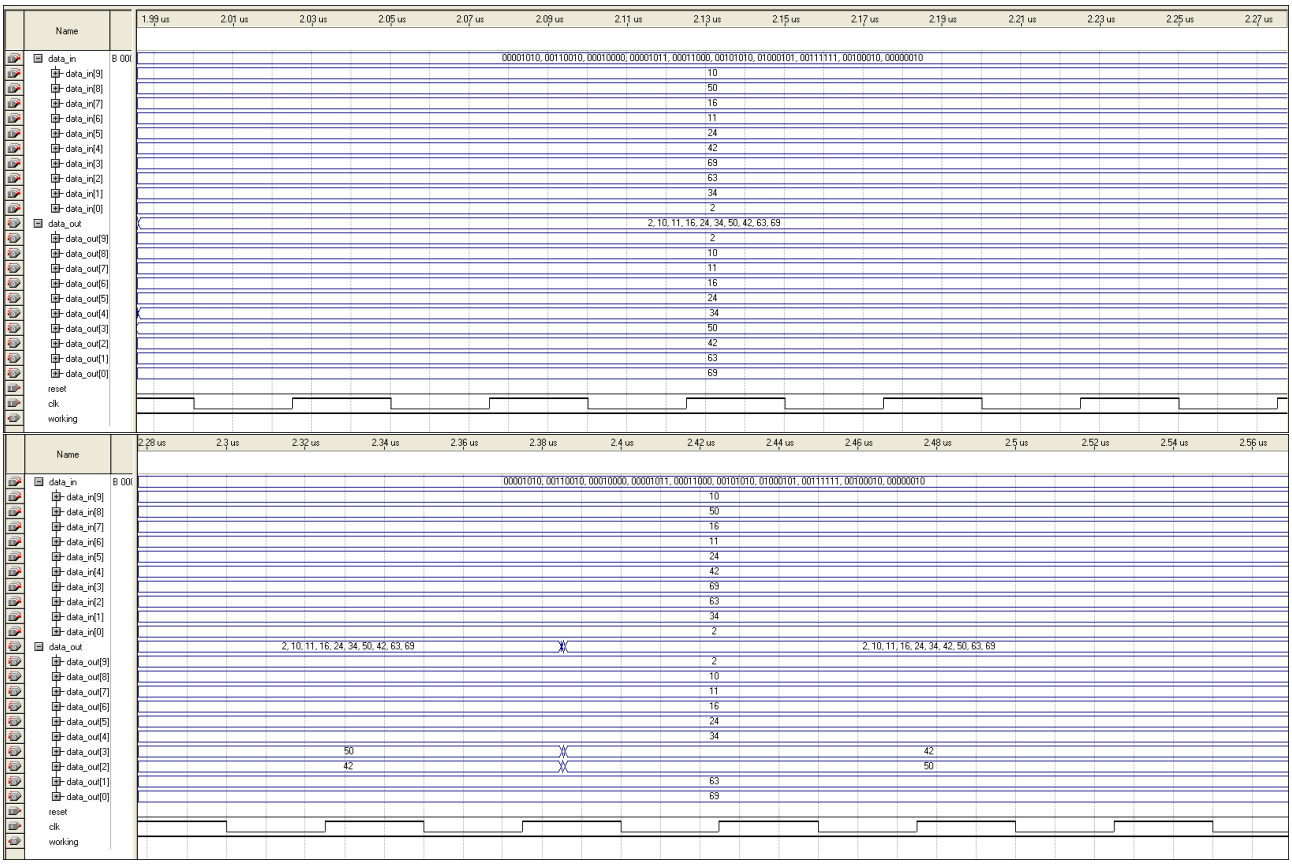
3.2 Симуляция (тестирование)

Waveform – диаграммы



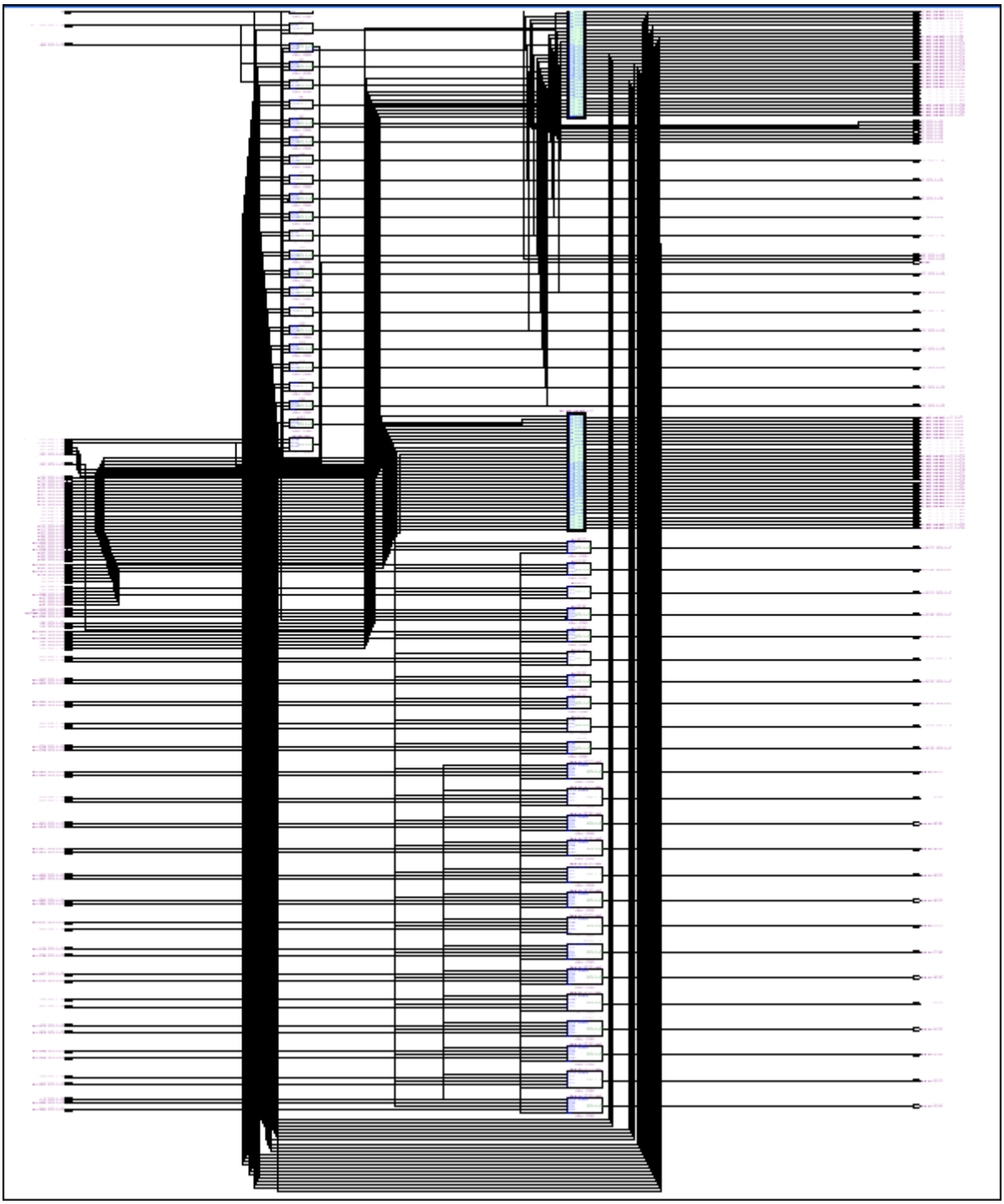
. . . . . .





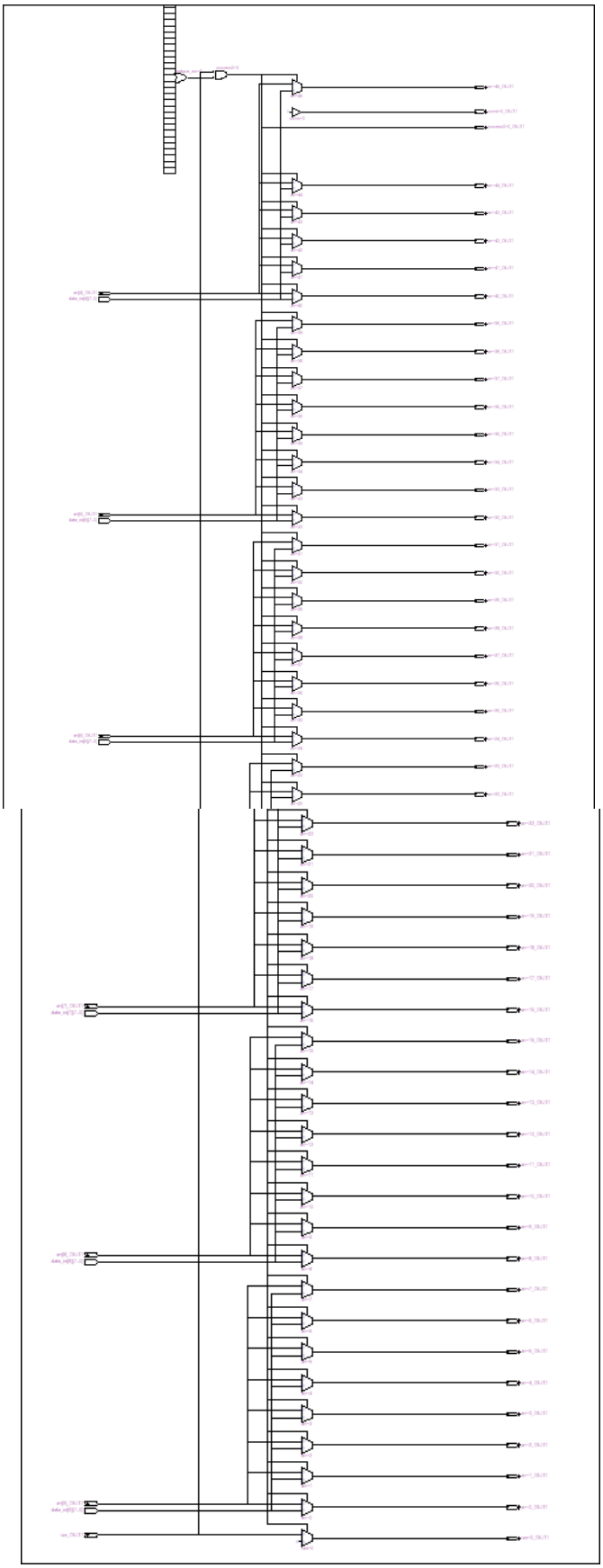
3.3 Технологическая схема

Технологическая схема создана на устройстве Stratex



3.4 RTL схема

RTL схема создана на устройстве Stratex



3.5 Отчёт в среде Quartus II

Flow Status	Successful - Mon Mar 25 21:58:52 2019
Quartus II Version	5.0 Build 148 04/26/2005 SJ Full Version
Revision Name	sort
Top-level Entity Name	sort
Family	FLEX10K
Met timing requirements	Yes
Total logic elements	761 / 1,152 (66 %)
Total pins	163 / 189 (86 %)
Total memory bits	0 / 12,288 (0 %)
Device	EPF10K20RC240-3
Timing Models	Final

Analysis & Synthesis Resource Usage Summary

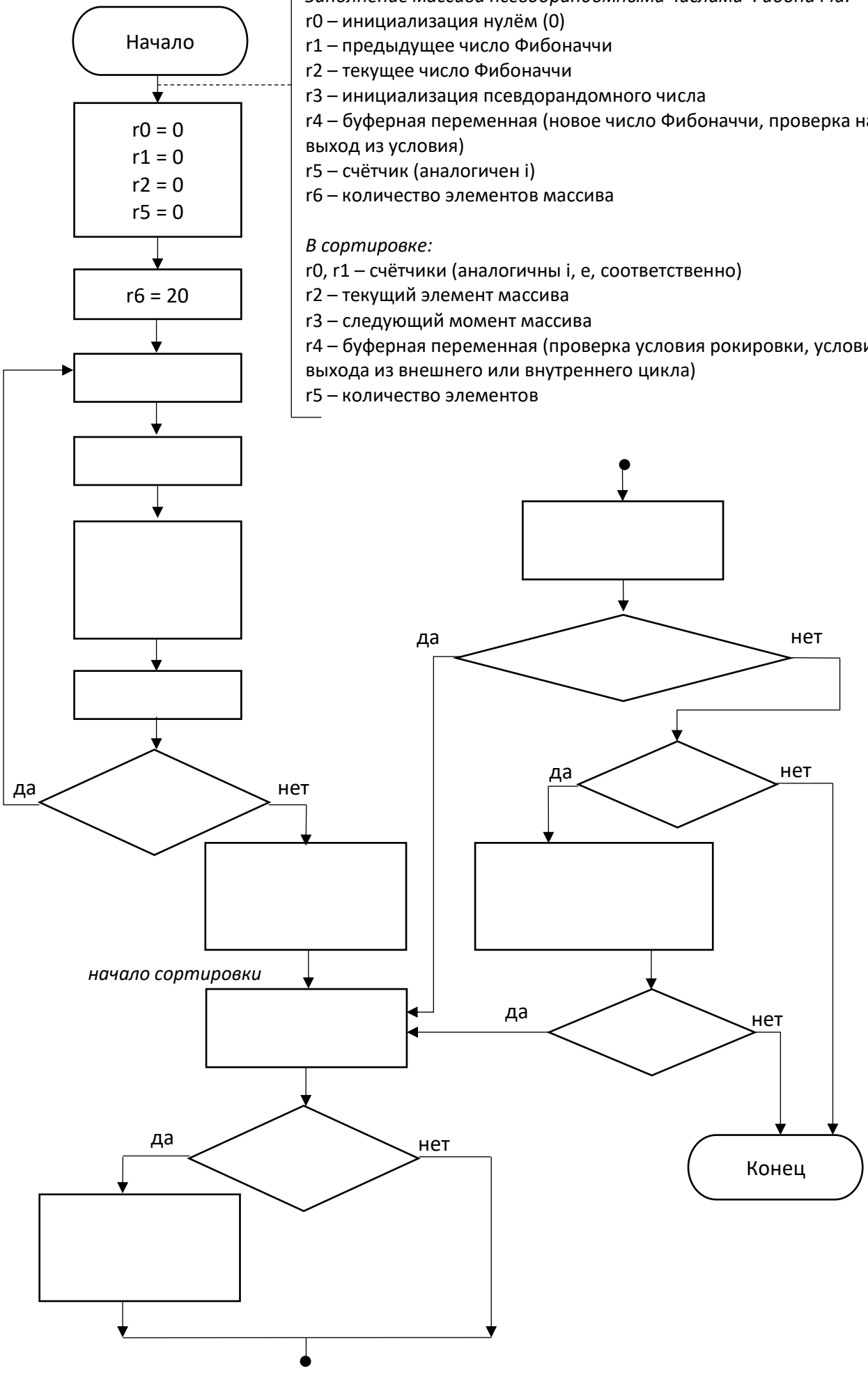
	Resource	Usage	
1	Total logic elements	761	
2	Total combinational functions	759	
3	-- Total 4-input functions	408	
4	-- Total 3-input functions	245	
5	-- Total 2-input functions	75	
6	-- Total 1-input functions	31	
7	-- Total 0-input functions	0	
8	Combinational cells for routing	0	
9	Total registers	226	
10	Total logic cells in carry chains	64	
11	I/O pins	163	
12	Maximum fan-out node	process0~323	
13	Maximum fan-out	242	
14	Total fan-out	3092	
15	Average fan-out	3.35	

4 Программная реализация алгоритма на VHDL-моделях DP32 и памяти

4.1 Блок схема

Заполнение массива псевдорандомными числами Фибоначчи:
r0 – инициализация нулём (0)
r1 – предыдущее число Фибоначчи
r2 – текущее число Фибоначчи
r3 – инициализация псевдорандомного числа
r4 – буферная переменная (новое число Фибоначчи, проверка на выход из условия)
r5 – счётчик (аналогичен i)
r6 – количество элементов массива

В сортировке:
r0, r1 – счётчики (аналогичны i, e, соответственно)
r2 – текущий элемент массива
r3 – следующий момент массива
r4 – буферная переменная (проверка условия рокировки, условия выхода из внешнего или внутреннего цикла)
r5 – количество элементов



4.2 Исполняемая программа в машинном коде

```
variable mem :memory_array:= —27000ns array
(X"0700_0000", —r0=0
X"1006_0014", —r6=20 arr_size
X"1001_0000", —r1=0 fibb_0
X"1002_0001", —r2=1 fibb_1
X"1003_0028", —r3=40 rand[0]
X"1004_0000", —r4=0 tmp
X"1005_0001", —r5=0 i
—loop
X"1203_0379", —r3*=121 (rand[i-1]*A) —endloop
…….
X"1006_0000", —r6=0 (e)
X"1000_0001", —r0=1 i
X"1001_0000", —r1=1 j

—loop

—loop
X"3002_0128", —r2<-M[r1+(?kolkom?)]
……
X"500B_00F5", —if(1=V&0|N&1|Z&1) then PC<-PC-10
—endloop
……
X"1006_0600", —r6=r6
—endloop
……..
X"5000_00FF", —if true then PC<-PC-1
—infinite loop
others =>X"0000_0000");
```

4.3 Исполняемая программа на языке ассемблера DP32

```
-- randomizer + fibonacci
r0 <- r0 & !(r0)
r6 <- r0 + 20
r1 <- r0 + 0
r2 <- r0 + 1
r3 <- r0 + 40
r4 <- r0 + 0
r5 <- r0 + 0
        -- loop
……..
        M[r3+42] <- r2
        r4 <- r1 + r2
…….
        if(1=V&0|N&1|Z&1) then PC <- PC - 12

-- bubble sort
……
        -- loop
                -- loop
                r2 <- M[r1 + 40]
                ……
                if(0=V&0|N&1|Z&0) then PC <- PC + 3
                        M[r1 + 40] <- r3
```

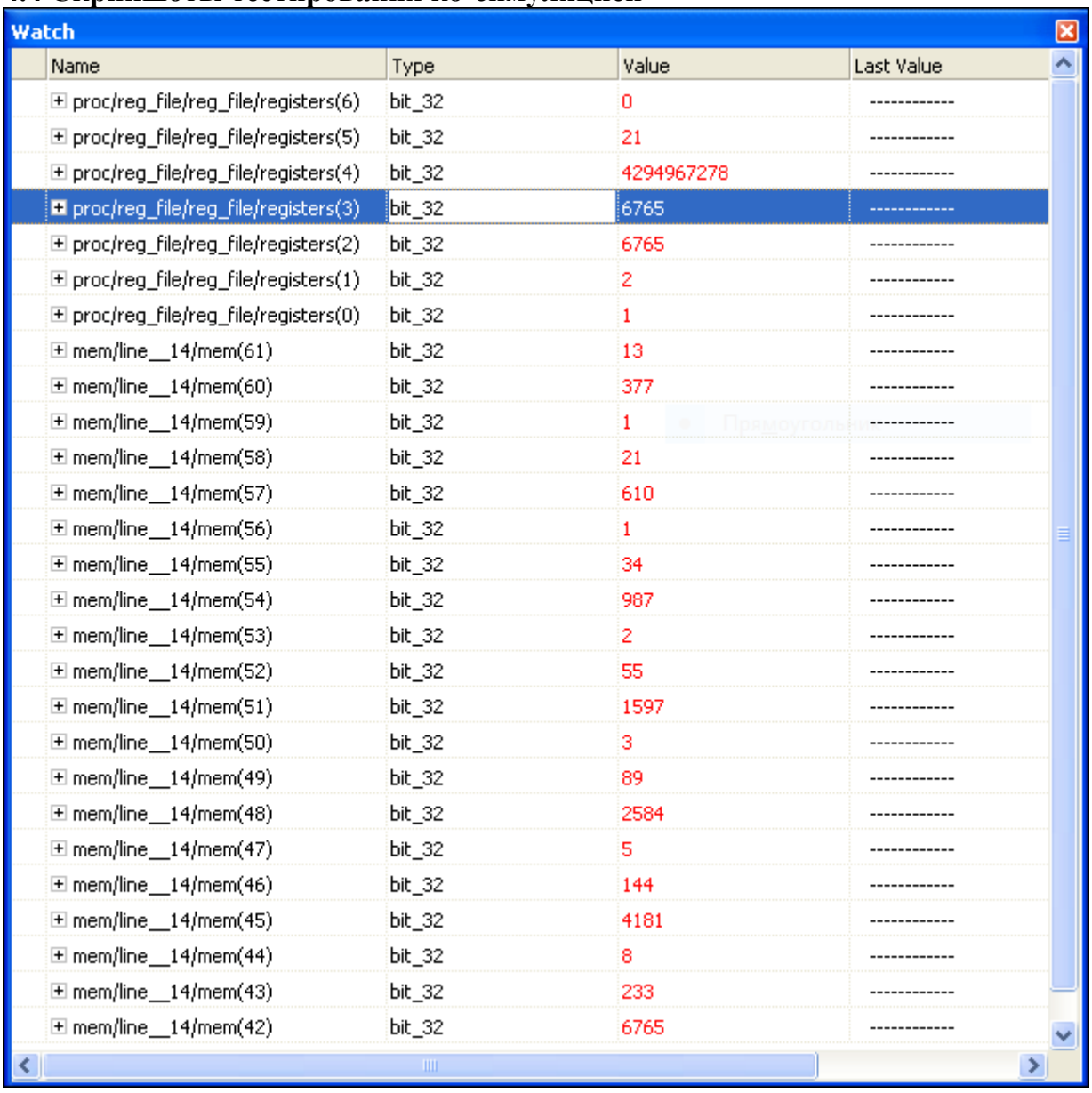
M[r1 + 41] <- r2

……

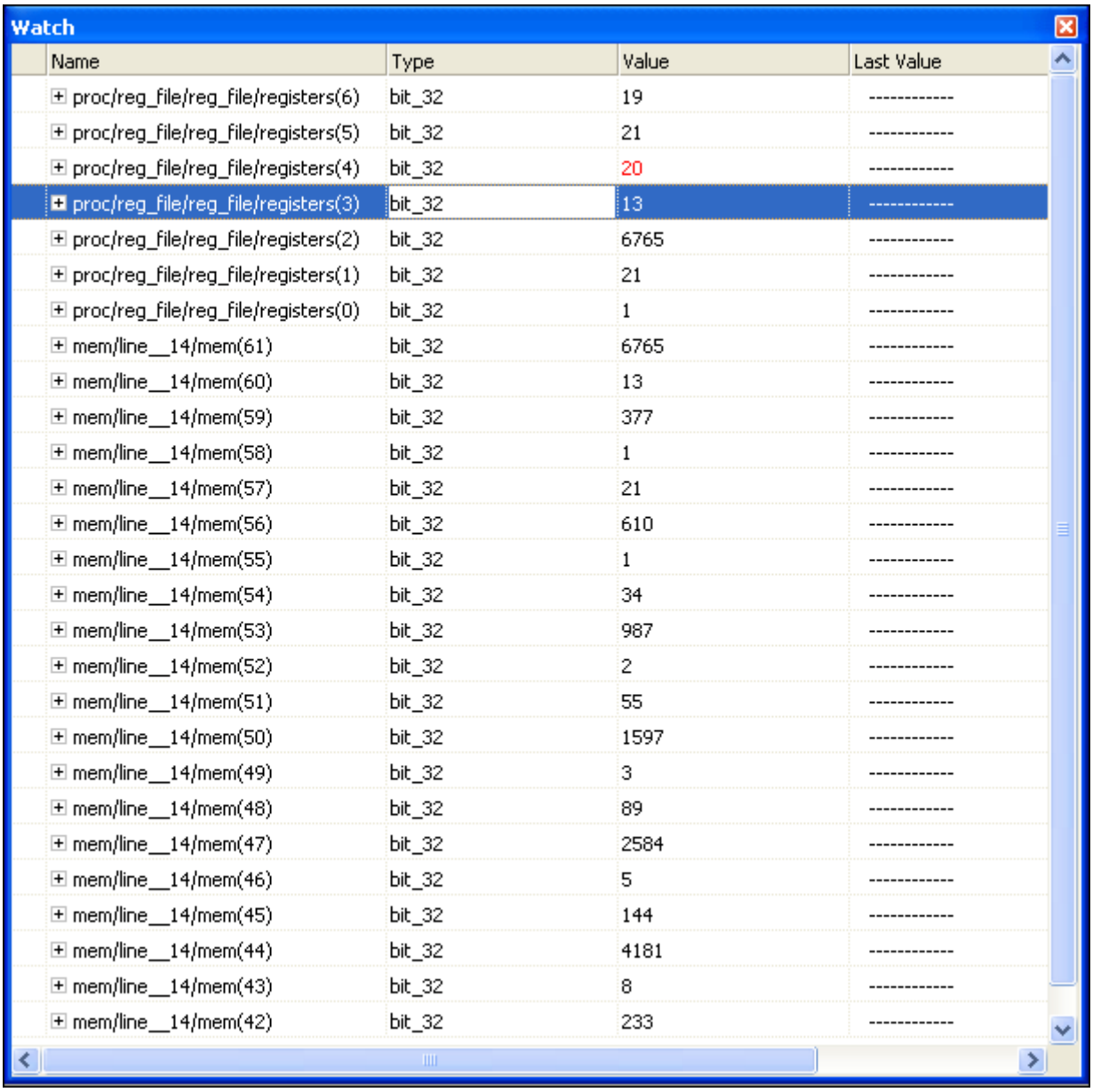
if(0=V&0|N&0|Z&1) then PC <- PC - 17

-- infinite loop
if true then PC <- PC - 1

4.4 Скриншоты тестирования ко-симуляцией



Массив не отсортирован



Сортировка. Итерация 1

Сортировка. Итерация 2

Сортировка. Итерация 3

Сортировка. Итерация 4

Сортировка. Итерация 5

Сортировка. Итерация 6

Сортировка. Итерация 7

Сортировка. Итерация 8

Сортировка. Итерация 9

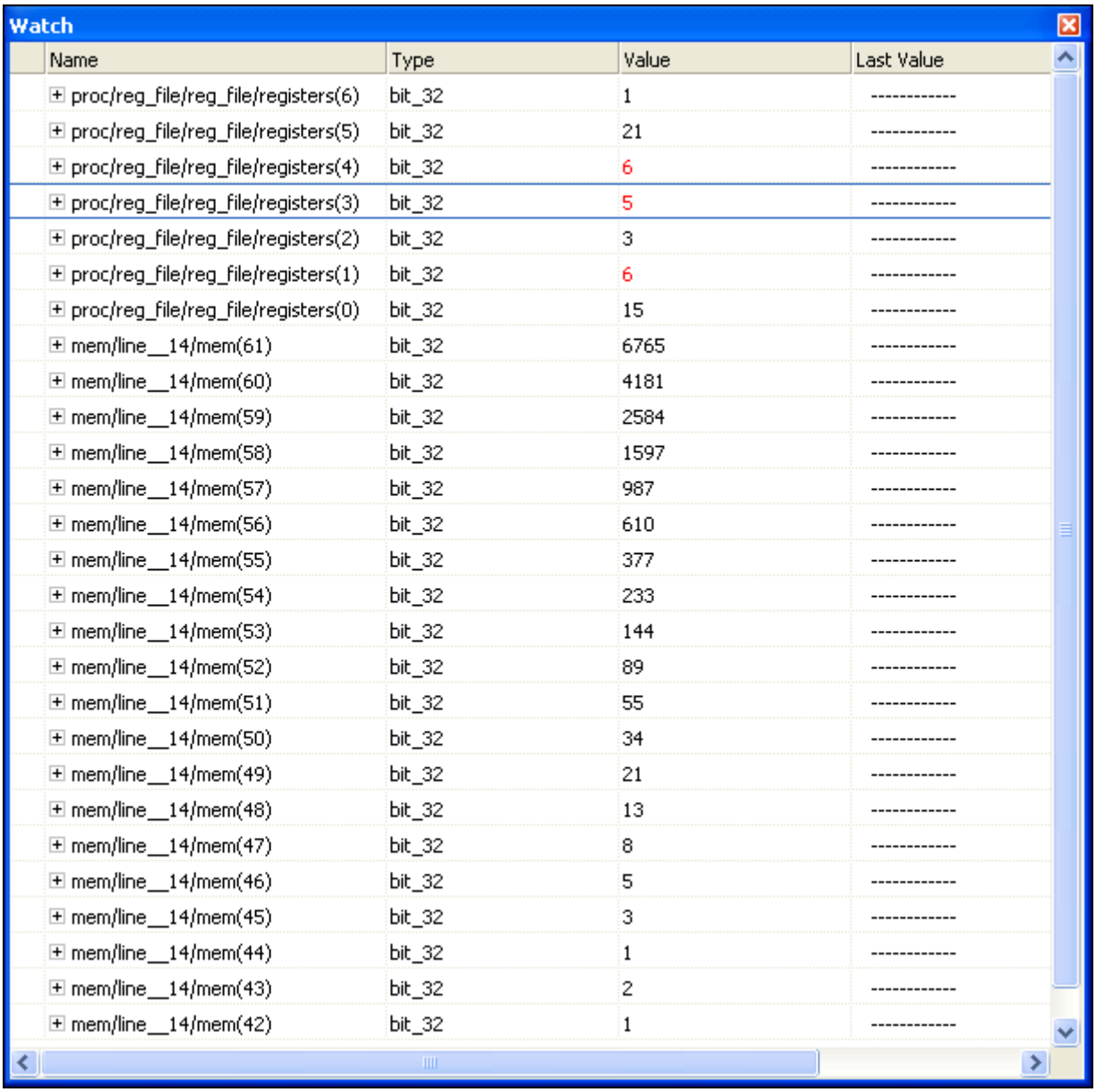
Сортировка. Итерация 10

Сортировка. Итерация 11

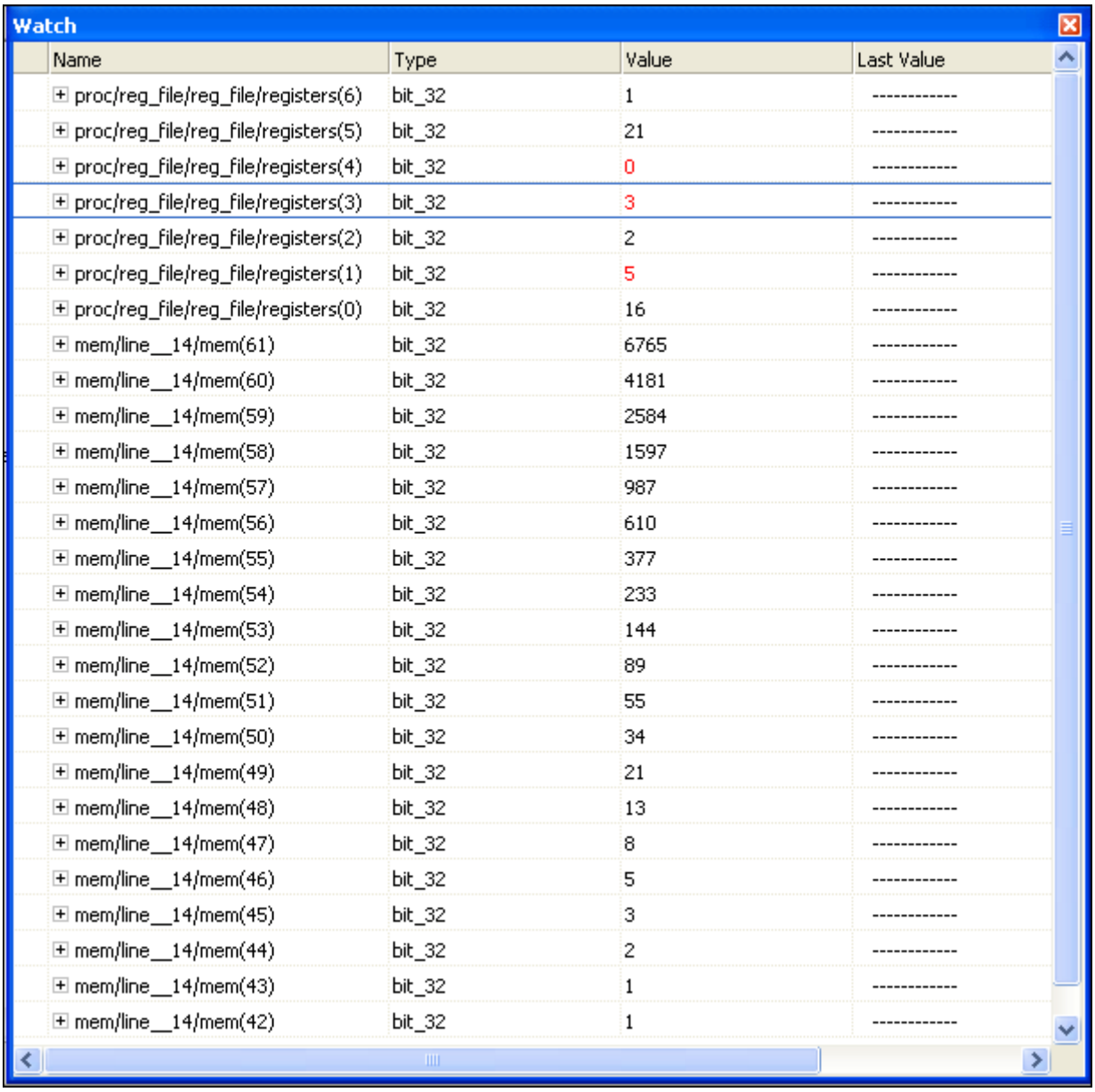
Сортировка. Итерация 12

Сортировка. Итерация 13

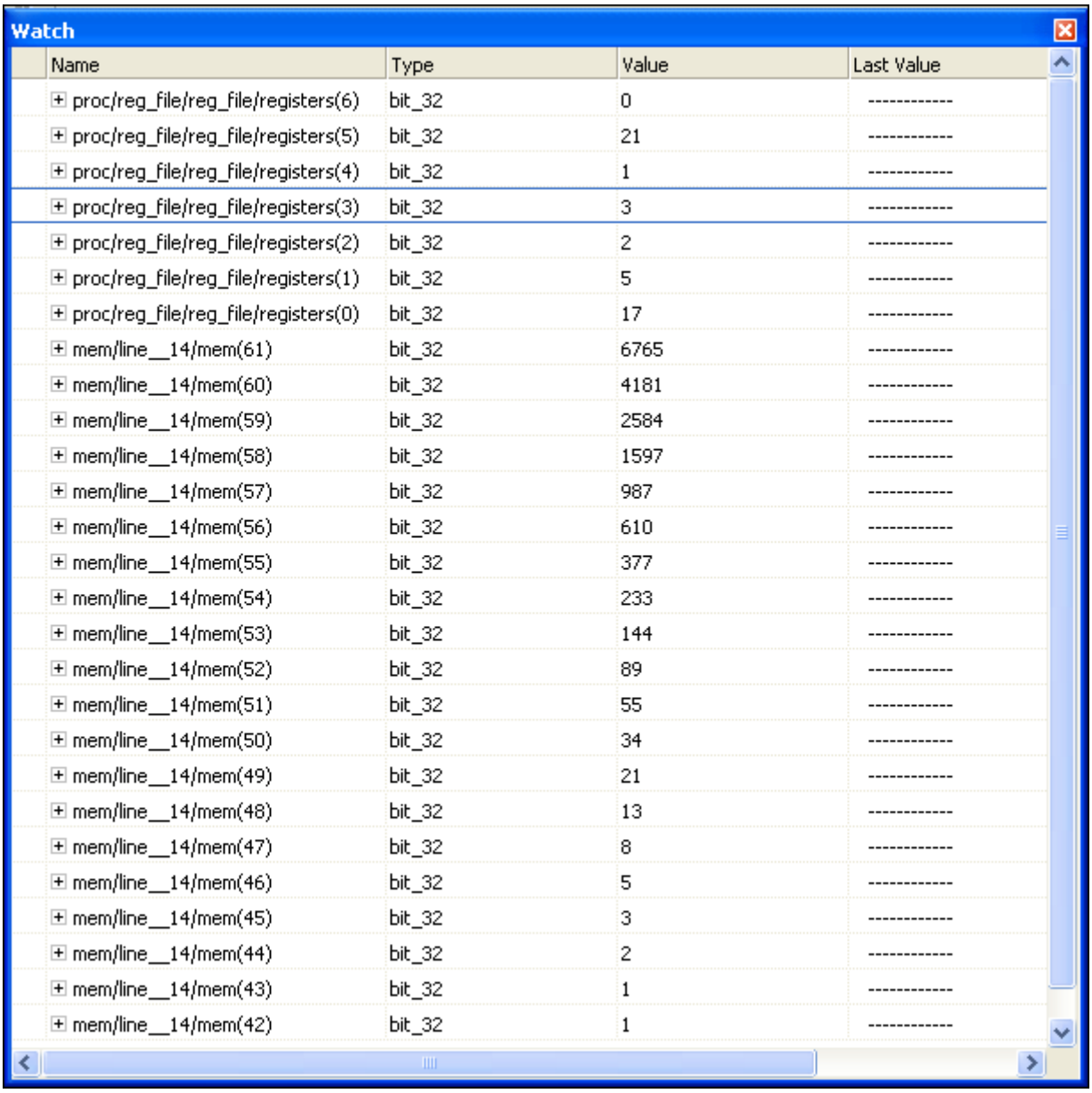
Сортировка. Итерация 14



Сортировка. Итерация 15



Массив отсортирован и представляет собой первые 20 элементов последовательности Фибоначчи



Программа вошла в вечный цикл (5000_00FF)

4.5 Ручной расчёт

Исходные данные: Массив из восьми чисел (4, 7, 3, 0, 2, 1, 8, 5)

- рассматриваемые элементы для рокировки;
подчеркнутые элементы – отсортированная часть массива

Работа алгоритма:
(4, 7, 3, 0, 2, 1, 8, 5)
(4, 3, 7, 0, 2, 1, 8, 5)
(4, 3, 0, 7, 2, 1, 8, 5)
(4, 3, 0, 2, 7, 1, 8, 5)
(4, 3, 0, 2, 1, 7, 8, 5)
(4, 3, 0, 2, 1, 7, 8, 5)
(4, 3, 0, 2, 1, 7, 5, 8)
(3, 4, 0, 2, 1, 7, 5, 8)
(3, 0, 4, 2, 1, 7, 5, 8)
(3, 0, 2, 4, 1, 7, 5, 8)
(3, 0, 2, 1, 4, 7, 5, 8)
(3, 0, 2, 1, 4, 7, 5, 8)
(3, 0, 2, 1, 4, 5, 7, 8)
(3, 0, 2, 1, 4, 5, 7, 8)
(0, 3, 2, 1, 4, 5, 7, 8)
(0, 2, 3, 1, 4, 5, 7, 8)
(0, 2, 1, 3, 4, 5, 7, 8)
(0, 2, 1, 3, 4, 5, 7, 8)
(0, 2, 1, 3, 4, 5, 7, 8)
(0, 2, 1, 3, 4, 5, 7, 8)
(0, 2, 1, 3, 4, 5, 7, 8)
(0, 2, 1, 3, 4, 5, 7, 8)
(0, 1, 2, 3, 4, 5, 7, 8)
(0, 1, 2, 3, 4, 5, 7, 8)
(0, 1, 2, 3, 4, 5, 7, 8)
(0, 1, 2, 3, 4, 5, 7, 8)
(0, 1, 2, 3, 4, 5, 7, 8)
(0, 1, 2, 3, 4, 5, 7, 8)