**Отчёт по лабораторной работе**

по дисциплине

«Языки моделирования и описания цифровой аппаратуры»

**Программная и аппаратная реализация алгоритма сортировки вставками**

Студент гр. 5130904/30008 Ребдев П.А

Студент гр. 5130904/30008 Мунгой Шеллер Валмиро Да Линда

Проверил: Амосов В.В

Оглавление

[1. Техническое задание 3](#__RefHeading___Toc163_1041169204)

[2. Описание алгоритма 3](#__RefHeading___Toc175_1041169204)

[3. Аппаратная реализация 4](#__RefHeading___Toc177_1041169204)

[3.1 VHDL код 4](#__RefHeading___Toc179_1041169204)

[3.2 Симуляция (тестирование) 5](#__RefHeading___Toc181_1041169204)

[3.3 RTL схема 6](#__RefHeading___Toc183_1041169204)

[3.4 Отчёт в среде Quartus II 6](#__RefHeading___Toc185_1041169204)

[4 Программная реализация алгоритма на VHDL-моделях DP32 и памяти 7](#__RefHeading___Toc187_1041169204)

[4.1 Блок схема 7](#__RefHeading___Toc189_1041169204)

[4.2 Исполняемая программа в машинном коде 8](#__RefHeading___Toc191_1041169204)

[4.3 Исполняемая программа на языке ассемблера DP32 9](#__RefHeading___Toc536_3299441894)

[4.4 Скриншоты тестирования ко-симуляцией 10](#__RefHeading___Toc195_1041169204)

[5. Реализация алгоритма на языке программирования C++ 12](#__RefHeading___Toc197_1041169204)

[6. Ручной расчёт 14](#__RefHeading___Toc199_1041169204)

## 1. Техническое задание

1) Разработать аппаратную реализацию сортировки вставсками 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) Протестировать алгоритм на высокоуровневом языке программирования

9) Сделать ручной расчёт сортировки

## 2. Описание алгоритма

На вход алгоритма подаётся последовательность n чисел. Алгоритм состоит из прохода по массиву. На каждом шагу следующий элемент сравнивается с текущим, если порядок неверный, то следующий элемент вставляется в отсортированную часть. Проход по массиву проводиться однократно, на каждом шаге отсортированная часть увеличивается, а не отсортированная уменьшается

## 3. Аппаратная реализация

### 3.1 VHDL код

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.std\_logic\_arith.all;

use ieee.std\_logic\_unsigned.all;

package types is

type integer\_array is array (natural range <>) of integer;

end package types;

package body types is

end package body types;

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.std\_logic\_arith.all;

use ieee.std\_logic\_unsigned.all;

use work.types.all;

entity insertSort is

generic

(

elementsNum : integer := 8

);

port

(

clk : in std\_logic;

start : in std\_logic;

reset : in std\_logic;

inArr : in integer\_array(0 to (elementsNum - 1));

outArr : out integer\_array(0 to (elementsNum - 1));

done : out std\_logic;

working : out std\_logic

);

end insertSort;

architecture behav of insertSort is

signal blocking : std\_logic;

begin

process(clk)

variable localArr : integer\_array(0 to (elementsNum - 1)) := (others => 0);

variable temp : integer := 0;

variable i : integer range 0 to (elementsNum - 1) := 0;

variable j : integer range -1 to (elementsNum - 1) := 0;

begin

if (rising\_edge(clk)) then

if (reset = '1') then

outArr <= (others => 0);

done <= '0';

blocking <= '1';

elsif(start = '1') then

outArr <= (others => 0);

done <= '0';

blocking <= '0';

localArr := inArr;

i := 0;

j := 0;

elsif (blocking = '0') then

if (i < (elementsNum - 1)) t

hen

if ((j >= 0) and (localArr(j) > localArr(j + 1))) then

temp := localArr(j);

localArr(j) := localArr(j + 1);

localArr(j + 1) := temp;

j := j - 1;

else

j := i + 1;

i := i + 1;

end if;

else

done <= '1';

outArr <= localArr;

blocking <= '0';

end if;

end if;

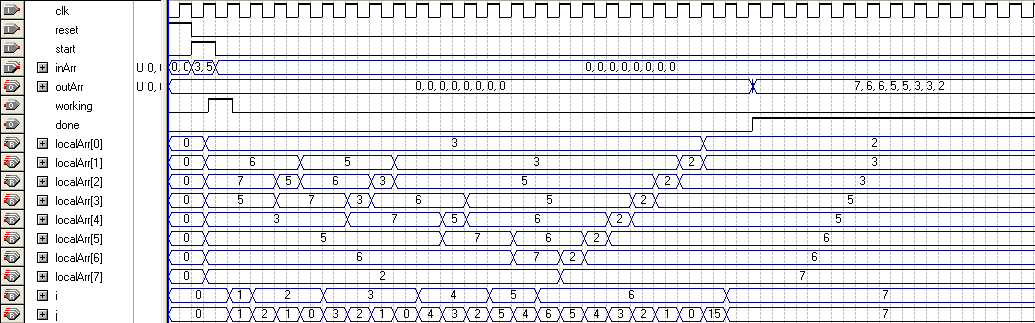
working <= blocking;

end if;

end process;

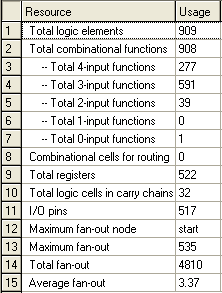
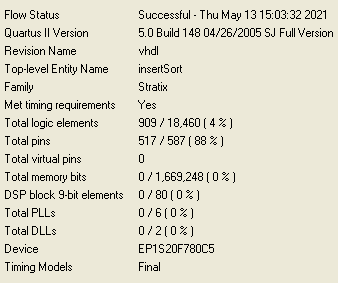
end behav;

### 3.2 Симуляция (тестирование)



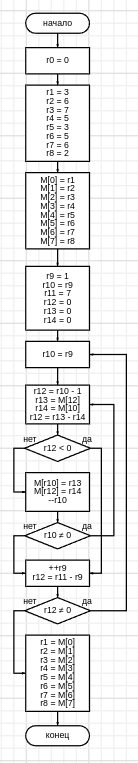
### 3.3 RTL схема

### 3.4 Отчёт в среде Quartus II



## 4 Программная реализация алгоритма на VHDL-моделях DP32 и памяти

### 4.1 Блок схема



### 4.2 Исполняемая программа в машинном коде

X"0700\_0000", -- r0 = 0;

-- Start array value

X"1001\_0003", -- r1 = 3;

X"1002\_0006", -- r2 = 6;

X"1003\_0007", -- r3 = 7;

X"1004\_0005", -- r4 = 5;

X"1005\_0003", -- r5 = 3;

X"1006\_0005", -- r6 = 5;

X"1007\_0006", -- r7 = 6;

X"1008\_0002", -- r8 = 2;

--Put start value's to memory

X"3101\_0000", -- M[0] = r1

X"3102\_0001", -- M[1] = r2

X"3103\_0002", -- M[2] = r3

X"3104\_0003", -- M[3] = r4

X"3105\_0004", -- M[4] = r5

X"3106\_0005", -- M[5] = r6

X"3107\_0006", -- M[6] = r7

X"3108\_0007", -- M[7] = r8

-- i and j

X"1009\_0001", -- r9(i) = 1;

X"000A\_0900", -- r10(j) = r9(i);

-- other variable

X"100B\_0008", -- r11(number of elements) = 8;

X"100C\_0000", -- r12(valToSwap/j - 1) = 0;

X"100D\_0000", -- r13(M[j - 1]) = 0;

X"100E\_0000", -- r14(M[j]) = 0;

-- main for: for(r9(i) = 1; r9(i) <= r11(number of elements); ++r9(i));

X"000A\_0900", -- r10(j) = r9(i)

-- local for: for(r10(j) = r9(i); r10(j) > 0; --r10(j));

-- if

X"110C\_0A01", -- r12 = r10(j) - 1

X"300D\_0C00", -- r13 = M[r12(j - 1)]

X"300E\_0A00", -- r14 = M[r10(j)]

X"010C\_0D0E", -- r12(valToSwap) = r13(M[j - 1]) - r14(M[j])

X"500A\_0005", -- если r12 < 0, выходим из local for

-- swap

X"310D\_0A00", -- M[r10(j)] = r13

X"110C\_0A01", -- r12 = r10(j) - 1

X"310E\_0C00", -- M[r12(j - 1)] = r14

X"110A\_0A01", -- --r10(j);

X"5001\_00F6", -- если j != 0, возращаемся в начало local for

X"1009\_0901", -- ++r9(i);

X"010C\_0B09", -- r12 = r11(number of elements) - r9(i) + 1

X"5001\_00F1", -- если r12 != 0, возращаемся в начало main for

--End of programm, save data to registers

X"3001\_0000", -- r1 = M[0]

X"3002\_0001", -- r2 = M[1]

X"3003\_0002", -- r3 = M[2]

X"3004\_0003", -- r4 = M[3]

X"3005\_0004", -- r5 = M[4]

X"3006\_0005", -- r6 = M[5]

X"3007\_0006", -- r7 = M[6]

X"3008\_0007", -- r8 = M[7]

X"100F\_0010", -- r15 = 16 (end of programm)

### [4.3 Исполняемая программа на языке ассемблера DP32](#__RefHeading___Toc193_1041169204)

r0 = r0 & !r0

-- Start array value

r1 = r0 + 3

r2 = r0 + 6

r3 = r0 + 7

r4 = r0 + 5

r5 = r0 + 3

r6 = r0 + 5

r7 = r0 + 6

r8 = r0 + 2

--Put start value's to memory

M[r0 + 0] = r1

M[r0 + 1] = r2

M[r0 + 2] = r3

M[r0 + 3] = r4

M[r0 + 4] = r5

M[r0 + 5] = r6

M[r0 + 6] = r7

M[r0 + 7] = r8

-- i and j

r9 = r0 + 1

r10 = r0 + r9

-- other variable

r11 = r0 + 7

r12 = r0 + 0

r13 = r0 + 0

r14 = r0 + 0

-- main for: for(r9(i) = 1; r9(i) <= r11(number of elements); ++r9(i));

r10 = r0 + r9

-- local for: for(r10(j) = r9(i); r10(j) > 0; --r10(j))

-- if

r12 = r10 - 1

r13 = M[r12 + 0]

r14 = M[r10 + 0]

r12 = r13 — r14

(V & 0) | (N & 1) | (Z & 0) = 1 Then PC <- PC + 5

-- swap

M[r10 + 0] = r13

r12 = r10 - 1

M[r12 + 0] = r14

r10 = r10 — 1

(V & 0) | (N & 0) | (Z & 1) = 0 Then PC <- PC - 11

r9 = r9 + 1

r12 = r11 — r9

(V & 0) | (N & 0) | (Z & 1) = 0 Then PC <- PC - 16

--End of programm, save data to registers

r1 = M[r0 + 0]

r2 = M[r0 + 1]

r3 = M[r0 + 2]

r4 = M[r0 + 3]

r5 = M[r0 + 4]

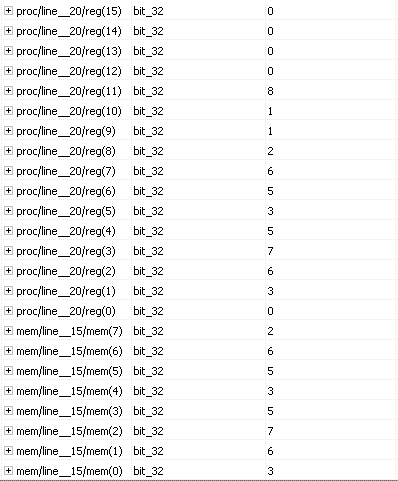
r6 = M[r0 + 5]

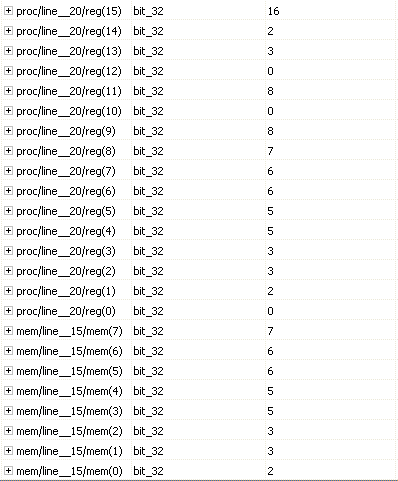
r7 = M[r0 + 6]

r8 = M[r0 + 7]

r15 = r0 + 16

### 4.4 Скриншоты тестирования ко-симуляцией

Состояние регистров и памяти перед «main for»

Финальное состояние программы

## 5. Реализация алгоритма на языке программирования C++

Для реализации алгоритма на высокоуровневом языке был выбран язык программирования c++.

main.cpp:

#include <iostream>

int main()

{

const int arrSize = 8;

int arr[arrSize];

std::cout << "\033[1;32mGenerate " << arrSize << " random numbers:\033[0m\n";

for (int i = 0; i < arrSize; ++i)

{

arr[i] = (std::rand() % 10);

std::cout << arr[i] << ' ';

};

std::cout << "\n\n";

for (int i = 0; i < (arrSize - 1); ++i)

{

std::cout << (i + 1) << ".\t";

for (int x : arr) std::cout << x << ' ';

std::cout << '\n';

int j = i;

while ((arr[j] > arr[j + 1]) && (j >= 0))

{

int c = arr[j];

arr[j] = arr[j + 1];

arr[j + 1] = c;

std::cout << (i + 1) << '.' << (i - j + 1) << ".\t";

for (int x : arr) std::cout << x << ' ';

std::cout << '\n';

--j;

}

}

std::cout << "\n\033[1;32mResult of sorting:\033[0m\n";

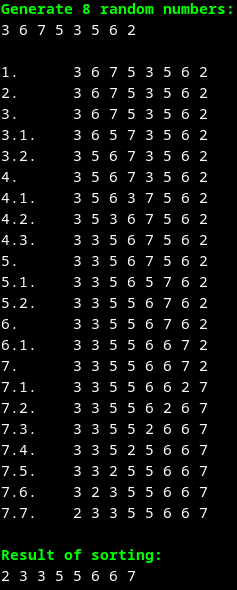
for (int x : arr) std::cout << x << ' ';

std::cout << '\n';

return 0;

}

Результат работы:



## 6. Ручной расчёт

Возьмём 8 случайно сгенерированых тестовых чисел: 3 6 7 5 3 5 6 2

