Федеральное государственное автономное образовательное учреждение
высшего образования «Санкт-Петербургский политехнический университет
Петра Великого»
Институт компьютерных наук и технологий
Программная инженерия

ПОЛИТЕХ
Санкт-Петербургский
политехнический университет
Петра Великого

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

Санкт-Петербург
2025

Оглавление

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

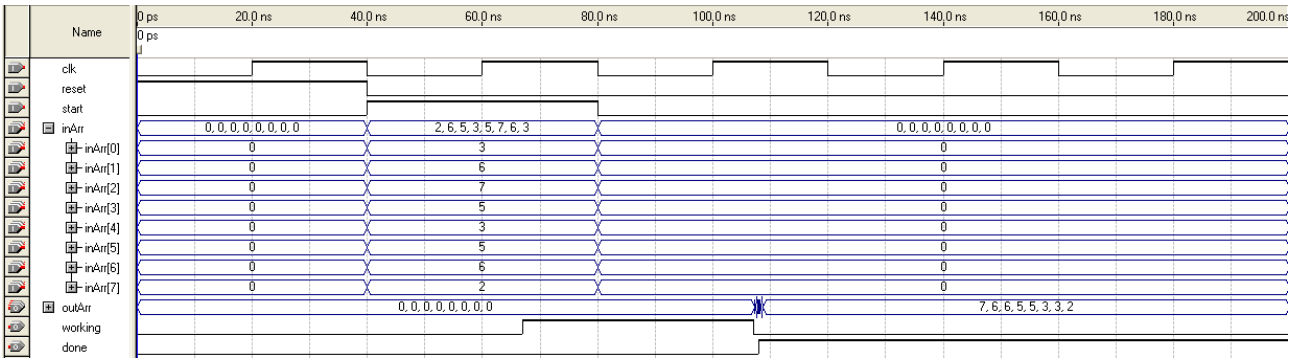
3. Аппаратная реализация

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;
    elsif (blocking = '0') then
      for i in 0 to (elementsNum - 2) loop
        j := i;
        while ((j >= 0) and (localArr(j) > localArr(j + 1))) loop
          temp := localArr(j);
          localArr(j) := localArr(j + 1);
          localArr(j + 1) := temp;
          j := j - 1;
        end loop;
      end loop;
      done <= '1';
      outArr <= localArr;
      blocking <= '0';
    end if;
    working <= blocking;
  end if;
 end process;
end behav;
```

3.2 Симуляция (тестирование)



3.3 RTL схема

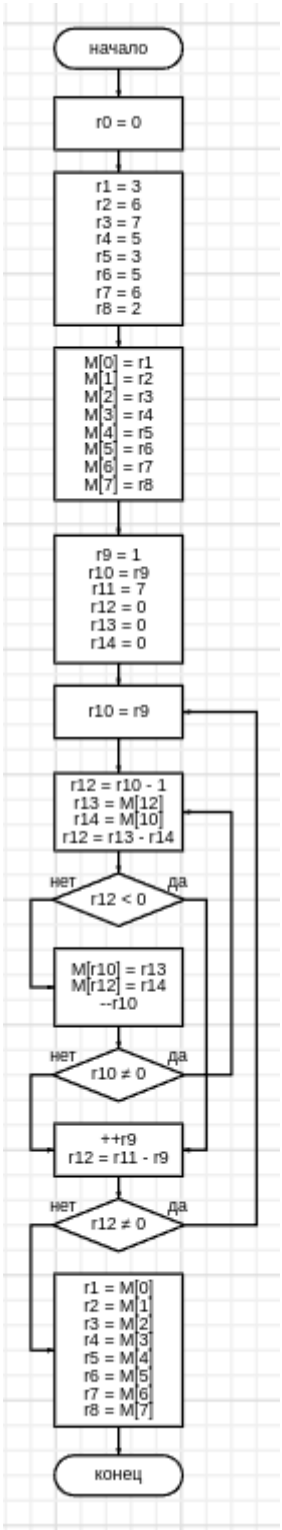
3.4 Отчёт в среде Quartus II

Flow Status	Successful - Thu Apr 29 15:02:22 2021
Quartus II Version	5.0 Build 148 04/26/2005 SJ Full Version
Revision Name	vhdl
Top-level Entity Name	insertSort
Family	Stratix
Met timing requirements	Yes
Total logic elements	2,965 / 18,460 (16 %)
Total pins	517 / 587 (88 %)
Total virtual pins	0
Total memory bits	0 / 1,669,248 (0 %)
DSP block 9-bit elements	0 / 80 (0 %)
Total PLLs	0 / 6 (0 %)
Total DLLs	0 / 2 (0 %)
Device	EP1S20F780C5
Timing Models	Final

	Resource	Usage
1	Total logic elements	2965
2	Total combinational functions	2964
3	-- Total 4-input functions	932
4	-- Total 3-input functions	1127
5	-- Total 2-input functions	904
6	-- Total 1-input functions	0
7	-- Total 0-input functions	1
8	Combinational cells for routing	0
9	Total registers	515
10	Total logic cells in carry chains	896
11	I/O pins	517
12	Maximum fan-out node	start
13	Maximum fan-out	516
14	Total fan-out	11587
15	Average fan-out	3.33

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, возвращаемся 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, возвращаемся 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

```
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

⊞ proc/line__20/reg(15)	bit_32	0
⊞ proc/line__20/reg(14)	bit_32	0
⊞ proc/line__20/reg(13)	bit_32	0
⊞ proc/line__20/reg(12)	bit_32	0
⊞ proc/line__20/reg(11)	bit_32	8
⊞ proc/line__20/reg(10)	bit_32	1
⊞ proc/line__20/reg(9)	bit_32	1
⊞ proc/line__20/reg(8)	bit_32	2
⊞ proc/line__20/reg(7)	bit_32	6
⊞ proc/line__20/reg(6)	bit_32	5
⊞ proc/line__20/reg(5)	bit_32	3
⊞ proc/line__20/reg(4)	bit_32	5
⊞ proc/line__20/reg(3)	bit_32	7
⊞ proc/line__20/reg(2)	bit_32	6
⊞ proc/line__20/reg(1)	bit_32	3
⊞ proc/line__20/reg(0)	bit_32	0
⊞ mem/line__15/mem(7)	bit_32	2
⊞ mem/line__15/mem(6)	bit_32	6
⊞ mem/line__15/mem(5)	bit_32	5
⊞ mem/line__15/mem(4)	bit_32	3
⊞ mem/line__15/mem(3)	bit_32	5
⊞ mem/line__15/mem(2)	bit_32	7
⊞ mem/line__15/mem(1)	bit_32	6
⊞ mem/line__15/mem(0)	bit_32	3

Состояние регистров и памяти перед «main for»

proc/line__20/reg(15)	bit_32	16
proc/line__20/reg(14)	bit_32	2
proc/line__20/reg(13)	bit_32	3
proc/line__20/reg(12)	bit_32	0
proc/line__20/reg(11)	bit_32	8
proc/line__20/reg(10)	bit_32	0
proc/line__20/reg(9)	bit_32	8
proc/line__20/reg(8)	bit_32	7
proc/line__20/reg(7)	bit_32	6
proc/line__20/reg(6)	bit_32	6
proc/line__20/reg(5)	bit_32	5
proc/line__20/reg(4)	bit_32	5
proc/line__20/reg(3)	bit_32	3
proc/line__20/reg(2)	bit_32	3
proc/line__20/reg(1)	bit_32	2
proc/line__20/reg(0)	bit_32	0
mem/line__15/mem(7)	bit_32	7
mem/line__15/mem(6)	bit_32	6
mem/line__15/mem(5)	bit_32	6
mem/line__15/mem(4)	bit_32	5
mem/line__15/mem(3)	bit_32	5
mem/line__15/mem(2)	bit_32	3
mem/line__15/mem(1)	bit_32	3
mem/line__15/mem(0)	bit_32	2

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:

```
Generate 8 random numbers:
3 6 7 5 3 5 6 2

1.      3 6 7 5 3 5 6 2
2.      3 6 7 5 3 5 6 2
3.      3 6 7 5 3 5 6 2
3.1.    3 6 5 7 3 5 6 2
3.2.    3 5 6 7 3 5 6 2
4.      3 5 6 7 3 5 6 2
4.1.    3 5 6 3 7 5 6 2
4.2.    3 5 3 6 7 5 6 2
4.3.    3 3 5 6 7 5 6 2
5.      3 3 5 6 7 5 6 2
5.1.    3 3 5 6 5 7 6 2
5.2.    3 3 5 5 6 7 6 2
6.      3 3 5 5 6 7 6 2
6.1.    3 3 5 5 6 6 7 2
7.      3 3 5 5 6 6 7 2
7.1.    3 3 5 5 6 6 2 7
7.2.    3 3 5 5 6 2 6 7
7.3.    3 3 5 5 2 6 6 7
7.4.    3 3 5 2 5 6 6 7
7.5.    3 3 2 5 5 6 6 7
7.6.    3 2 3 5 5 6 6 7
7.7.    2 3 3 5 5 6 6 7

Result of sorting:
2 3 3 5 5 6 6 7
```

6. Ручной расчёт

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