**Міністерство освіти і науки України**

**Національний університет «Львівська політехніка»**

**Кафедра ЕОМ**


# Звіт

# до лабораторної роботи № 3

з дисципліни «Комп’ютерні системи»
на тему: «Аналіз програмної моделі процесу роботи арифметичного конвеєра, ч.1.»

Варіант №1

Виконав:

ст.гр. КІ-38

Александрова Р.-Є.О.

Прийняв:

Старший викладач

Козак Н. Б.

**Львів 2022**

**Мета:** Навчитись здійснювати аналіз програмних моделей комп'ютерних систем, виконаних на мові System C.

**Завдання:**

1. Проаналізувати склад програмної моделі арифметичного конвеєра, (програма PIPE), яка виконана на мові System C.
2. Визначити інформаційні потоки у моделі арифметичного конвеєра.
3. Визначити зв'язки керування.
4. Накреслити блоки, з яких складається арифметичний конвеєр згідно поданої моделі.
5. Оформити звіт.

**Хід роботи:**

У даній програмі визначено 5 блоків арифметичного конвеєра. Перший блок Numgen має один вхід для синхроімпульсу(clk) та два виходи(out1, out2). Даний блок генерує значення вхідних даних.

Другим блоком є блок обчислення (stage1). У нього наявні три вхідні(in1, in2, clk) та два вихідні(sum, diff) порти. На вхідні порти поступають синхроімпульс(вхід clk) та два вхідні сигнали, з якими будуть виконуватися обчислення. У результаті роботи даного блоку на виході sum буде подана сума поданих на входи in1 та in2 значень, на виході diff – різниця поданих на входи in1 та in2 значень.

Третій блок(stage2) має таку ж кількість входів та виходів, як і другий(три вхідні порти: clk, sum, diff та два вихідні порти: prod, quot). На вхід clk подається синхроімпульс, на входи sum, diff – вхідні значення. У результаті роботи даного блоку обчислення на виході prod буде результат множення вхідних значень, на виході quot – результат ділення вхідних значень.

У четвертому блоці описано три вхідні порти(clk, prod, qout) та один вихідний(powr). На вхід clk надходить синхроімпульс, на входи prod, qout – вхідні значення, над якими будуть проводитись обчислення. На вихід powr, у результаті роботи даного блоку обчислення, буде подано результат піднесення значення на вході prod до степеня, який дорівнює значенню на вході qout. У випадку, якщо на обох входах буде значення менше нуля, на вихід буде подаватись значення 0.

П'ятим блоком є блок Display, який має два входи: вхід синхроімпульсу clk та вхід in. Даний блок виконує виведення результату обчислень на екран.

Згідно із описом усіх блоків, можна зробити висновок, що усі блоки є синхронізованими.

У головній функції (sc_main) описано з'єднання усіх блоків. Для їх з'єднання створено 8 сигналів(in1,in2, sum, diff, prod, quot, powr, clk). Блок Numgen під'єднаний до блоку stage1 сингалами in1, in2, блок stage1 з'єднаний з блоком stage2 сигналами sum, diff, до відповідних входів та виходів. Блок stage2

з’єднаний з блоком stage3 сигналами prod, quot. Блок stage3 з’єднаний з блоком display сигналом powr. Усі блоки з’єднані одним сигналом clk.

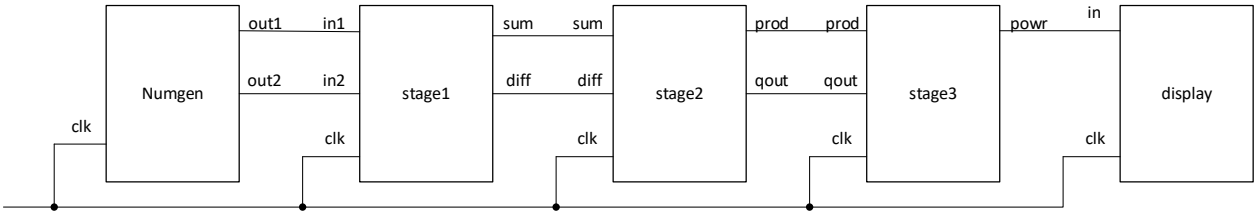


*Рис.1. Структурна схема тестового арифметичного конвеєра*

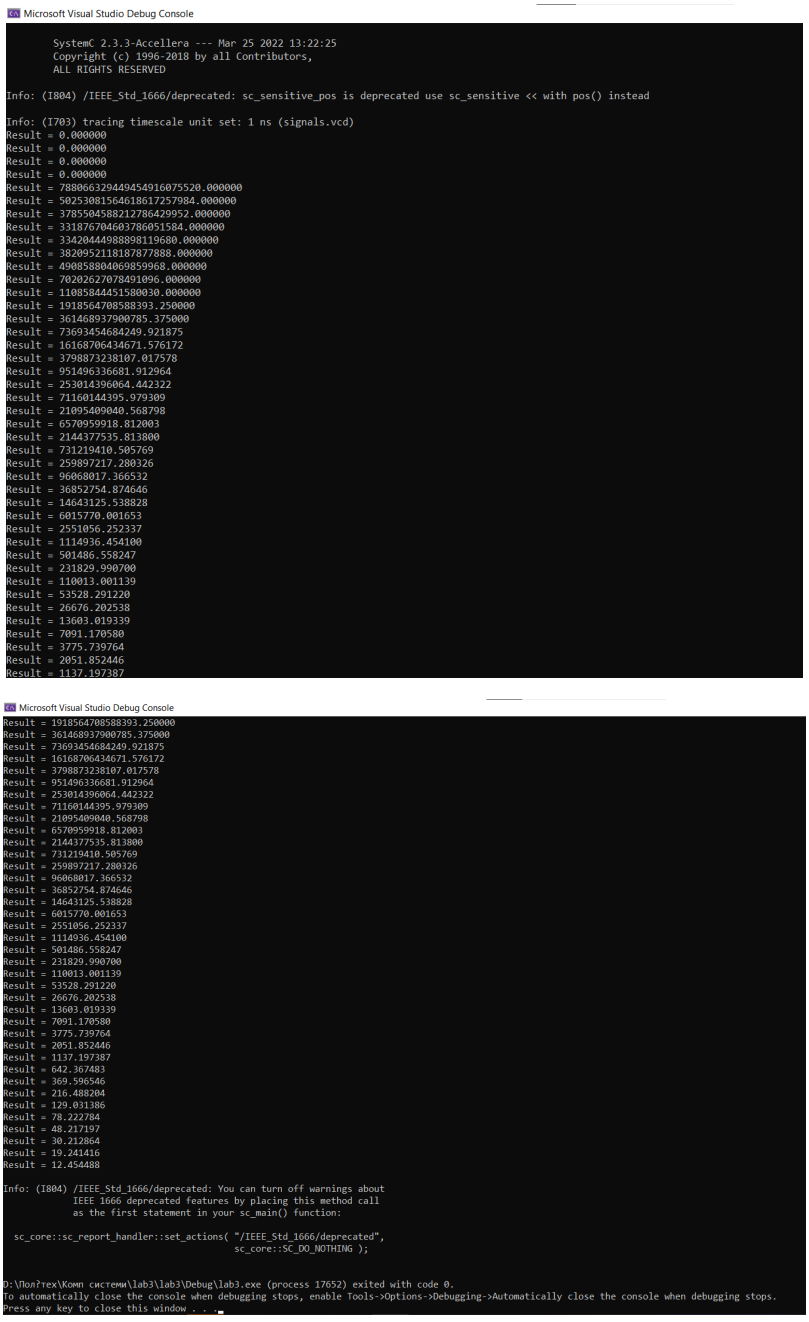


*Рис.2. Результат роботи програми*

Код програми:

```
#include "systemc.h"
#include "stage1.h"
#include "stage2.h"
#include "stage3.h"
#include "display.h"
#include "numgen.h"
#define NS * 1e-5
int sc_main(int ac, char* av[])
{
      //Signals
      sc_signal<double> in1;
      sc_signal<double> in2;
      sc_signal<double> sum;
      sc_signal<double> diff;
      sc_signal<double> prod;
      sc_signal<double> quot;
      sc_signal<double> powr;
      sc_signal<bool> clk;
      //Clock
      //<TRACE>
      //</TRACE>
      numgen N("numgen");                //instance of `numgen' module
      N(in1, in2, clk);                 //Positional port binding
      stage1 S1("stage1");               //instance of `stage1' module
      //Named port binding
      S1.in1(in1);
      S1.in2(in2);
      S1.sum(sum);
      S1.diff(diff);
      S1.clk(clk);

      sc_trace_file* wf = sc_create_vcd_trace_file("signals");
      wf->set_time_unit(1, SC_NS);
      sc_trace(wf, clk, "CLK");
      sc_trace(wf, in1, "In1");
      sc_trace(wf, in2, "In2");
      sc_trace(wf, sum, "Sum");
      sc_trace(wf, diff, "Diff");
      sc_trace(wf, prod, "Prod");
```

```
      sc_trace(wf, quot, "Quot");
      sc_trace(wf, powr, "Pow");

      stage2 S2("stage2");              //instance of `stage2' module
      S2(sum, diff, prod, quot, clk);  //Positional port binding
      stage3 S3("stage3");              //instance of `stage3' module
      S3(prod, quot, powr, clk);       //Positional port binding
      display D("display");            //instance of `display' module
      D(powr, clk);                     //Positional port binding
      //<TRACE>

      //</TRACE>
      //<TRACE>
      sc_start(0, SC_NS);
      for (int i = 0; i < 50; i++)
      {
            clk.write(0);
            sc_start(10, SC_NS);
            clk.write(1);
            sc_start(10, SC_NS);
      }
      sc_close_vcd_trace_file(wf);
      //</TRACE>
      return 0;
}
```

Numgen.h, numgen.cpp:

```
#include "systemc.h"
#ifndef NUMGEN_H
#define NUMGEN_H

struct numgen : sc_module {
      sc_out<double> out1;      //output 1
      sc_out<double> out2;      //output 2
      sc_in<bool>    clk;       //clock

      // method to write values to the output ports
      void generate();

      //Constructor
      SC_CTOR(numgen) {
```

```
            SC_METHOD(generate);   //Declare generate as SC_METHOD and
            sensitive_pos << clk;    //make it sensitive to positive clock
edge

      }
};

#endif
#include "systemc.h"
#include "numgen.h"

// definition of the `generate' method
void numgen::generate()
{
      static double a = 134.56;
      static double b = 98.24;
      a -= 1.5;
      b -= 2.8;
      out1.write(a);
      out2.write(b);

} // end of `generate' method

Stage1.h, stag1.cpp:
#ifndef STAGE1_H
#define STAGE1_H

struct stage1 : sc_module {

      sc_in<double> in1;   //input 1
      sc_in<double> in2;   //input 2
      sc_out<double> sum;  //output 1
      sc_out<double> diff; //output 2
      sc_in<bool>    clk;  //clock

      void addsub();       //method implementing functionality

      //Counstructor
      SC_CTOR(stage1) {
            SC_METHOD(addsub);   //Declare addsub as SC_METHOD and
```

```
          sensitive_pos << clk;   //make it sensitive to positive clock
edge
     }

public:

};

#endif
#include "systemc.h"
#include "stage1.h"
//Definition of addsub method
void stage1::addsub()
{
     double a;
     double b;
     a = in1.read();
     b = in2.read();
     sum.write(a + b);
     diff.write(a - b);
} // end of addsub method

Stage2.h, stage2.cpp
#include "systemc.h"
#ifndef STAGE2_H
#define STAGE2_H

struct stage2 : sc_module {
     sc_in<double>  sum;      //input port 1
     sc_in<double>  diff;     //input port 2
     sc_out<double> prod;     //output portik 1
     sc_out<double> quot;     //output portik 2
     sc_in<bool>    clk;      //clock


     void multdiv();          //method providing functionality

     //Constructor
     SC_CTOR(stage2) {
          SC_METHOD(multdiv);   //Declare multdiv as SC_METHOD and
```

```
            sensitive_pos << clk;    //make it sensitive to positive clock
edge.
      }

};
#endif
#include "systemc.h"
#include "stage2.h"
//definition of multdiv method
void stage2::multdiv()
{
      double a;
      double b;
      a = sum.read();
      b = diff.read();
      if (b == 0)
            b = 5.0;
      prod.write(a * b);
      quot.write(a / b);

} // end of multdiv

Stage3.h, stage3.cpp
#include "systemc.h"
#ifndef STAGE3_H
#define STAGE3_H

struct stage3 : sc_module {
      sc_in<double>  prod;      //input port 1
      sc_in<double>  quot;      //input port 2
      sc_out<double> powr;      //output port 1
      sc_in<bool>    clk;       //clock

      void power();             //method implementing functionality

      //Constructor
      SC_CTOR(stage3) {
            SC_METHOD(power);    //declare power as SC_METHOD and
            sensitive_pos << clk;   //make it sensitive to positive clock
edge
```

```
      }

};
#endif

#include "systemc.h"
#include "stage3.h"
void stage3::power()
{
      double a;
      double b;
      double c;

      a = prod.read();
      b = quot.read();
      c = (a > 0 && b > 0) ? pow(a, b) : 0.;
      powr.write(c);

} // end of power method

Display.h, display.cpp
#include "systemc.h"
#ifndef DISPLAY_H
#define DISPLAY_H

struct display : sc_module {
      sc_in<double> in;         // input port 1
      sc_in<bool>   clk;        // clock
      void print_result();           // method to display input port values

      //Constructor
      SC_CTOR(display) {
            SC_METHOD(print_result);    // declare print as SC_METHOD and
            sensitive_pos << clk;  // make it sensitive to positive clock
edge
      }

public:
};
#endif
```

```
#include "systemc.h"
#include "display.h"
#include <stdio.h>
//Definition of print_result method
void display::print_result()
{
     printf("Result = %f\n", in.read());
} // end of print method
```

**Висновок:** під час виконання даної лабораторної роботи було здійснено аналіз програмних моделей комп’ютерних систем, виконаних на мові System C.