**Міністерство освіти і науки України**

**Національний університет «Львівська політехніка»**



**Звіт**

про виконання лабораторної роботи №3

з дисципліни «Комп’ютерні системи»

На тему: " Аналіз програмної моделі процесу роботи арифметичного конвеєра, ч.1."

Виконав: ст. гр. КІ-38

Яців О.А.

Перевірив:

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.

**Хід виконання роботи:**

**Перелік і призначення блоків арифметичного конвеєра:**

Блок stage1 має два входи і обчислює суму вхідних даних та їх різницю.

Блок stage2 приймає дані з попереднього блоку та обчислює їх добуток та частку.

Блок stage3 приймає дані з попереднього блоку та обчислює перше вхідне число підняте до степені, що рівна другому вхідному числу

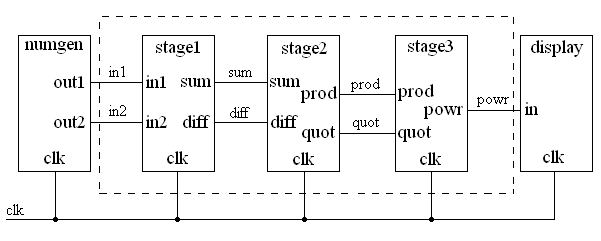
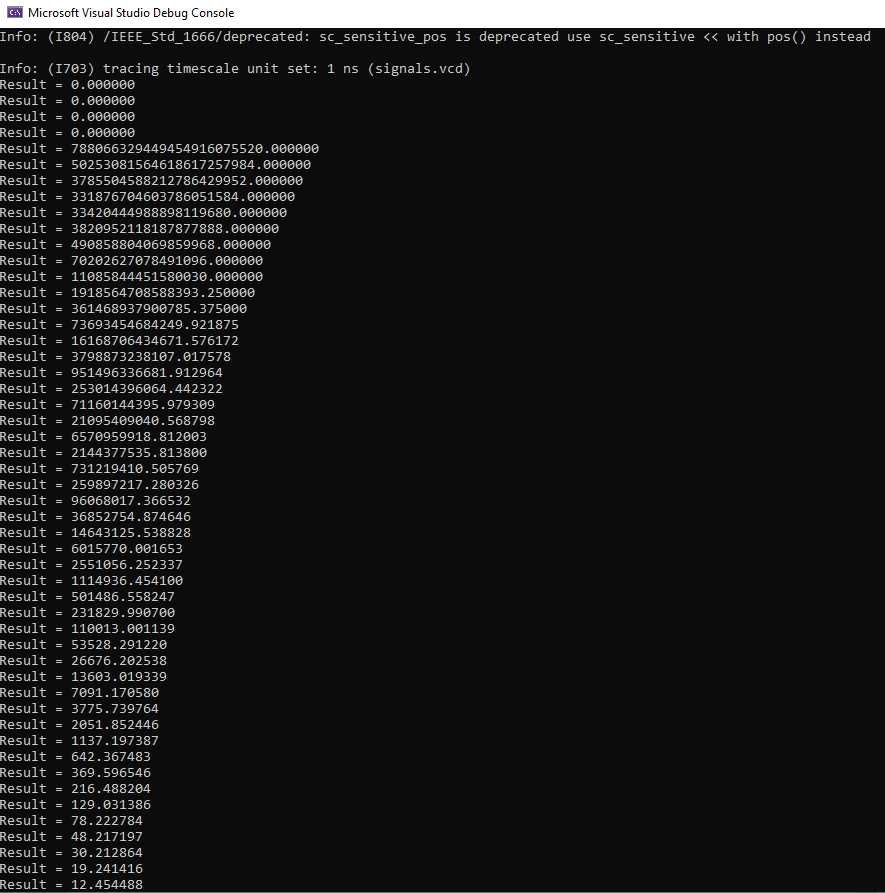
**Структурна схема арифметичного конвеєра, що відповідає програмній моделі:**

Рис.1. Структурна схема



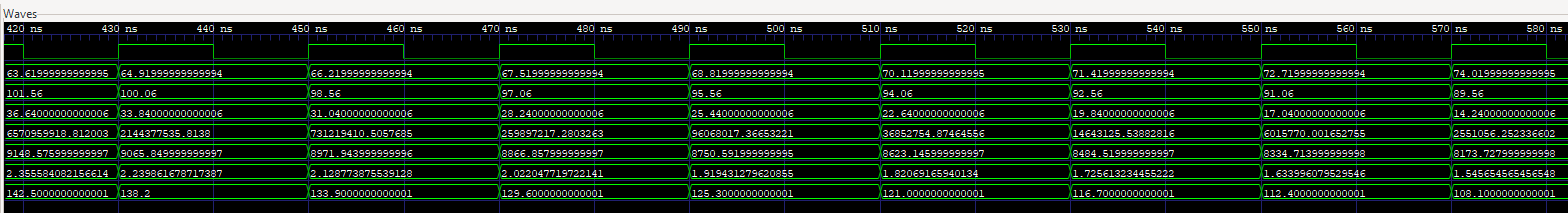
рис.2 виконання програми

рис.3 часова діаграма

**Виконання програми:**

**main.cpp**

#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, diff); //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;

}

**Display.cpp**

#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

**Display.h**

#include "systemc.h"

#ifndef DISPLAY\_H

#define DISPLAY\_H

//#include "stage1.h" // Added by ClassView

struct display : sc\_module {

sc\_in<double> in; // input port 1

sc\_in<bool> clk; // clock

sc\_in<double> diff;

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

**Stage1.cpp**

#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

**Stage1.h**

#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

**Stage2.cpp**

#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

**Stage2.h**

#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

**Stage3.cpp**

#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

**Stage3.h**

#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

**Numgen.cpp**

#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

**Numgen.h**

#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

**Висновок:** виконавши дану лабораторну роботу, я навчився здійснювати аналіз програмних моделей комп’ютерних систем, виконаних на мові System тC.