Міністерствоосвіти і науки України

Національний університет „Львівська політехніка”

Кафедра ЕОМ



**Лабораторна робота №4**

З дисципліни:”Комп’ютерні системи”

На тему:” **Аналіз програмної моделі процесу роботи арифметичного конвеєра, ч.2.**”

Виконав: ст.гр. КІ-32

Біленко В.М. Прийняв: Козак Н.Б.

Львів 2020

**МЕТА РОБОТИ**

Навчитись здійснювати аналіз програмних моделей комп’ютерних систем, виконаних на мові System C.

**Завдання до лабораторної**

1. Проаналізувати склад програмної моделі арифметичного конвеєра, (програма PIPE), яка виконана на мові System C.

2. Здійснити модернізацію функцій або параметрів арифметичного конвеєра (див. лабораторну роботу № 3), шляхом під’єднання розроблених модулів S1 та S2 (див. лабораторну роботу № 2). Порядок та тип з’єднання мають бути обгрунтовані, можливо розробка буферних або додаткових модулів з метою надавання нових властивостей тестувальній моделі.

3. Накреслити кінцеву структурну схему отриманої програмної моделі.

4. Навести стисло код та внесені нові зміни.

5. Навести результати тестування та використання програмної моделі.

**Аналіз програмної моделі:**

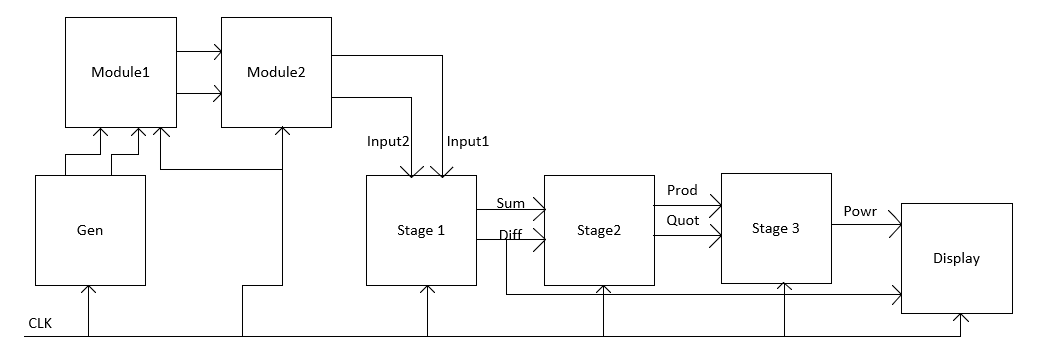


Рис.1. Блоки арифметричного конвеєра

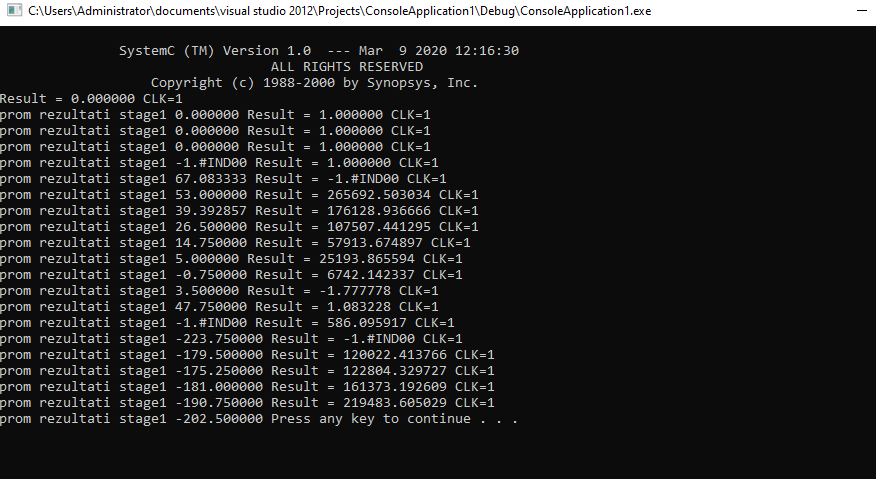


Рис.2.Результат роботи програми

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

#include "systemc.h"

#include "stage1.h"

#include "stage2.h"

#include "stage3.h"

#include "display.h"

#include "numgen.h"

#define NS \* 1e-9

struct module1 : sc\_module {

sc\_in<double> in1, in2; // input port 1

sc\_in<bool> clk; // clock

sc\_out<double> Res,F2;

void do\_add()

{

double Ain,Bin;

double temp = 1;

/\*double b;

a = in1.read();

b = in2.read();

sum.write(a+b);

diff.write(a-b);\*/

Ain = in1.read();

Bin = in2.read();

/\*for(double i = 1; i <= Ain;i++)

{

temp \*= i;

}\*/

Ain = Ain \* 4;

Res.write(Ain + Bin);

F2.write(Ain / Bin);

//F.write(~A.read() + B.read());

//F1.write(A.read() / B.read());

};

//Constructor

SC\_CTOR( module1 ) {

SC\_METHOD( do\_add ); // declare print as SC\_METHOD and

sensitive\_pos << clk; // make it sensitive to positive clock edge

}

public:

};

struct module2 : sc\_module {

sc\_in<double> F1, B; // input port 1

sc\_in<bool> clk; // clock

sc\_out<double> Res,F2;

void do\_next()

{

double Cin,Din;

double temp = 1;

Cin = F1.read();

Din = B.read();

/\*for(int i = 1; i <= Cin; i++)

{

temp \*= i;

}\*/

Cin = Cin / 2;

Res.write(Cin);

F2.write(Din + Cin\*Din);

//F.write(~A.read() + B.read());

//F1.write(A.read() / B.read());

};

//Constructor

SC\_CTOR( module2 ) {

SC\_METHOD( do\_next ); // declare print as SC\_METHOD and

sensitive\_pos << clk; // make it sensitive to positive clock edge

}

public:

};

int sc\_main(int ac, char \*av[])

{

//Signals

sc\_signal<double> in1;

sc\_signal<double> in2;

sc\_signal<double> Res;

sc\_signal<double> F2;

sc\_signal<double> mod2\_1;

sc\_signal<double> mod2\_2;

sc\_signal<double> sum;

sc\_signal<double> diff;

sc\_signal<double> prod;

sc\_signal<double> quot;

sc\_signal<double> powr;

//Clock

sc\_signal<bool> clk;

numgen N("numgen"); //instance of `numgen' module

N(in1, in2, clk ); //Positional port binding

// in1 = 4;

//in2 = 2;

module1 M1("module1");

M1(in1, in2, clk, Res, F2);

module2 M2("module2");

M2(Res, F2, clk, mod2\_1, mod2\_2);

// display D("display"); //instance of `display' module

//D(mod2\_1,clk, mod2\_2);

stage1 S1("stage1"); //instance of `stage1' module

//Named port binding

S1.in1(mod2\_2);

S1.in2(mod2\_1);

S1.sum(sum);

S1.diff(diff);

S1.clk(clk);

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

sc\_initialize(); //Initialize simulation

for(int i = 0; i < 20; i++){

clk.write(1);

sc\_cycle( 10 NS );

clk.write(0);

sc\_cycle( 10 NS );

}

system("pause");

return 0;

}

SC\_MODULE (stage1) {

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 << clk.pos(); //make it sensitive to positive clock edge

}

public:

};

SC\_MODULE (stage2) {

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 << clk.pos(); //make it sensitive to positive clock edge.

}

};

SC\_MODULE (stage3) {

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 << clk.pos(); //make it sensitive to positive clock edge

}

};

#if !defined(\_\_BCPLUSPLUS\_\_)

//what the heck is that, Amit? BCB complains. remove it

extern "C" double pow(double, double);

#endif

#endif

#ifndef DISPLAY\_H

#define DISPLAY\_H

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

SC\_MODULE (display){

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

sc\_in<bool> clk; // clock

sc\_in<double> diff;

void print(); // method to display input port values

//Constructor

SC\_CTOR(display) {

SC\_METHOD(print); // declare print as SC\_METHOD and

sensitive << clk; // make it sensitive to positive clock edge

}

public:

};

#ifndef NUMGEN\_H

#define NUMGEN\_H

SC\_MODULE (numgen) {

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 << clk.pos(); //make it sensitive to positive clock edge

}

}; #include "pch.h"

#include "systemc.h"

#include "stage1.h"

//Definition of addsub method

void stage1::addsub()

{

sum.write(in1.read() + in2.read());

diff.write(in1.read() - in2.read());

} // end of addsub method

#include "pch.h"

#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

//Definition of power method

void stage3::power()

{

double a;

double b;

double c;

a = prod.read();

b = quot.read();

c = pow(a, b);

powr.write(c);

} // end of power method

// 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

void display::print()

{

printf("Result = %f CLK=%d\n", in.read(), clk.read());

printf("prom rezultati stage1 %f ", diff.read());

} // end of print method