Міністерство освіти і науки України

Національний університет "Львівська політехніка"

Кафедра ЕОМ

****

**Звіт**

з лабораторної роботи № 3

з дисципліни: “ Комп’ютерні системи”

на тему: “ Аналіз програмної моделі процесу роботи арифметичного конвеєра, ч.1.”

Варіант 6

Виконав: студент .гр. КІ-32

Данилів М.М.

Прийняв: асистент каф. ЕОМ

Козак Н.Б.

Львів 2020

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

**ПОРЯДОК ВИКОНАННЯ РОБОТИ**

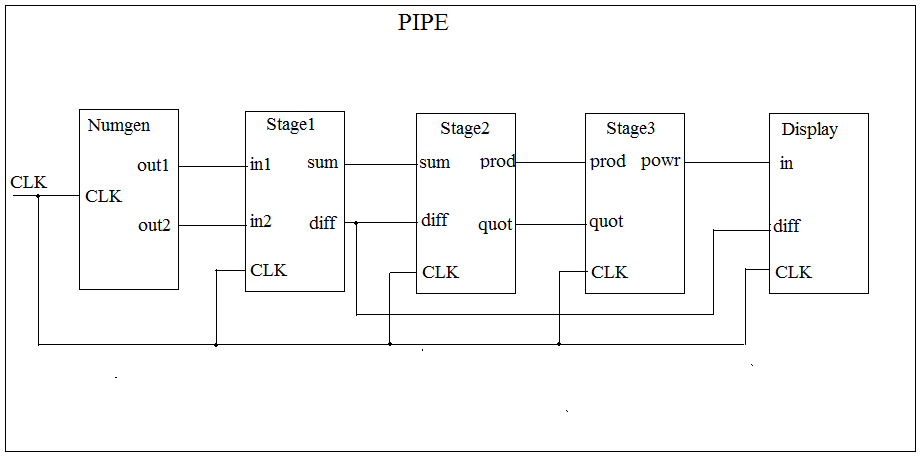
**Завдання до лабораторної**

а) дослідити лістинг програмної моделі арифметичного конвеєра на мові System C;

б) проаналізувати перелік і призначення блоків арифметичного конвеєра; провести дослідження та запуск тестової програмної моделі.

в) на базі проведених досліджень, підготувати структурну схему тестового арифметичного конвеєра;

г) навести стислий код, навести висновки.



**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();

//Counstructor

SC\_CTOR( stage1 ) {

SC\_METHOD( addsub );

sensitive\_pos << clk;

}

};

#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

**Stage2.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();

//Constructor

SC\_CTOR( stage2 ) {

SC\_METHOD( multdiv );

sensitive\_pos << clk; }

};

#endif

**Stage2.cpp**

#include "systemc.h"

#include "stage2.h"

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);

}

**Stage3.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();

//Constructor

SC\_CTOR( stage3 ){

SC\_METHOD( power

sensitive\_pos << clk; }

};

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

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

#endif

#endif

**Stage3.cpp**

#include <math.h>

#include "systemc.h"

#include "stage3.h"

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

**Numgen.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

void generate();

//Constructor

SC\_CTOR( numgen ) {

SC\_METHOD( generate );

sensitive\_pos << clk;

}

};

#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

**Display.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();

//Constructor

SC\_CTOR( display ) {

SC\_METHOD( print );

sensitive\_pos << clk;

}

};

#endif

**Display.cpp**

#include "systemc.h"

#include "display.h"

#include <stdio.h>

//Definition of print method

void display::print()

{

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

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

} // end of print method

**main.cpp**

#include "systemc.h"

#include "stage1.h"

#include "stage2.h"

#include "stage3.h"

#include "display.h"

#include "numgen.h"

#define NS \* 1e-9

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;

//Clock

sc\_signal<bool> clk;

numgen N("numgen");

N(in1, in2, clk );

stage1 S1("stage1");

S1.in1(in1);

S1.in2(in2);

S1.sum(sum);

S1.diff(diff);

S1.clk(clk);

stage2 S2("stage2");

S2(sum, diff, prod, quot, clk );

stage3 S3("stage3");

S3( prod, quot, powr, clk);

display D("display");

D(powr,clk,diff);

sc\_initialize();

for(int i = 0; i < 50; i++){

clk.write(1);

sc\_cycle( 10 NS );

clk.write(0);

sc\_cycle( 10 NS );

}

return 0;

}

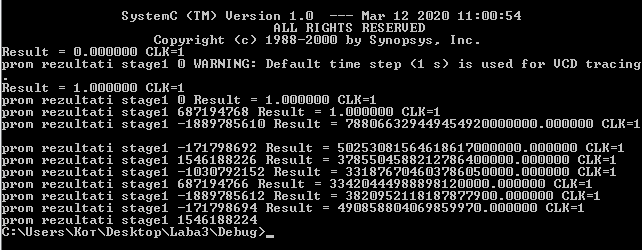


Рис.1 Результат виконання програми

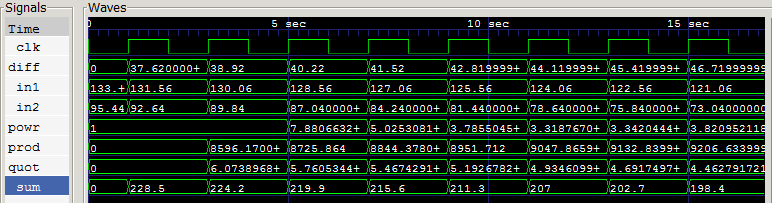


Рис.2 Графік сигналів

**Висновок:** На цій лабораторній роботі я ознайомився з основними конструкціями мови моделювання System C.