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 10

Виконав: студент .гр. КІ-33

Лялька О.О.

Прийняв: асистент каф. ЕОМ

Козак Н.Б.

Львів 2020

**Тема:** Аналіз програмної моделі процесу роботи арифметичного конвеєра.

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

**Індивідуальне завдання:** 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.

**Хід виконання роботи:**

1) Лістинг програмної моделі арифметичного конвеєра на мові System C

File

main.cpp

#include "systemc.h"

#include "stage1.h"

#include "stage2.h"

#include "stage3.h"

#include "S1.h"

#include "S2.h"

#include "display.h"

#include "numgen.h"

#define NS \* 1e-9

int sc\_main(int, char \*[]) {

//Signals

sc\_signal<double> in1;

sc\_signal<double> in2;

sc\_signal<double> sum;

sc\_signal<double> diff;

sc\_signal<double> prod;

sc\_signal<double> f1;

sc\_signal<double> r1;

sc\_signal<double> r2;

sc\_signal<double> rep;

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

S1 s\_1("s1");

s\_1(in1, in2, powr, f1, clk);

S2 s\_2("s2");

s\_2(f1, rep, r1, r2, clk);

stage1 S\_1("stage1"); //instance of `stage1' module

//Named port binding

S\_1.in1(r1);

S\_1.in2(r2);

S\_1.sum(sum);

S\_1.diff(diff);

S\_1.clk(clk);

stage2 S\_2("stage2"); //instance of `stage2' module

S\_2(sum, diff, prod, quot, clk); //Positional port binding

stage3 S\_3("stage3"); //instance of `stage3' module

S\_3(prod, quot, powr, clk); //Positional port binding

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

D(powr, clk); //Positional port binding

sc\_initialize(); //Initialize simulation

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

clk.write(1);

sc\_cycle(10 NS);

clk.write(0);

sc\_cycle(10 NS);

}

return 0;

}

}

*S1.h*

#ifndef TEST\_SYSTEMC\_S1\_H

#define TEST\_SYSTEMC\_S1\_H

#include "systemc.h"

#include "cmath"

#include "S2.h"

SC\_MODULE(S1) {

sc\_in<int> x;

sc\_in<int> y;

sc\_out<int> f1;

sc\_out<int> rep;

sc\_in<bool> clk;

SC\_CTOR(S1) {

SC\_METHOD(calc\_rep);

sensitive << x << y;

sensitive << clk.pos();

SC\_METHOD(calc\_f1);

sensitive << x << y;

sensitive << clk.pos();

}

void calc\_rep() {

int x\_val = x.read();

int y\_val = y.read();

double acc = 0;

acc = !(x\_val || y\_val);

rep.write(acc);

}

void calc\_f1() {

int x\_val = x.read();

int y\_val = y.read();

int acc = 0;

acc = x\_val || y\_val;

f1.write(acc);

}

};

#endif TEST\_SYSTEMC\_S1\_H

#pragma once

*S2.h*

#ifndef TEST\_SYSTEMC\_S2\_H

#define TEST\_SYSTEMC\_S2\_H

#include "systemc.h"

SC\_MODULE(S2) {

sc\_in<int> a;

sc\_in<int> b;

sc\_out<int> r1;

sc\_out<int> r2;

sc\_in<bool> clk;

SC\_CTOR(S2) {

SC\_METHOD(calc\_r1);

sensitive << a << b;

sensitive << clk.pos();

SC\_METHOD(calc\_r2)

sensitive << a << b;

sensitive << clk.pos();

}

void calc\_r1() {

int b\_val = b.read();

r1.write(b\_val /2);

}

void calc\_r2() {

int a\_val = a.read();

int b\_val = b.read();

r2.write(a\_val-b\_val);

}

};

#endif TEST\_SYSTEMC\_S2\_H

1. Розробив кінцеву структурну схему отриманої модифікованої програмної моделі:

#include "systemc.h"

#include "stage1.h"

#include "stage2.h"

#include "stage3.h"

#include "S1.h"

#include "S2.h"

#include "display.h"

#include "numgen.h"

#define NS \* 1e-9

int sc\_main(int, char \*[]) {

//Signals

sc\_signal<double> in1;

sc\_signal<double> in2;

sc\_signal<double> sum;

sc\_signal<double> diff;

sc\_signal<double> prod;

sc\_signal<double> f1;

sc\_signal<double> r1;

sc\_signal<double> r2;

sc\_signal<double> rep;

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

S1 s\_1("s1");

s\_1(in1, in2, powr, f1, clk);

S2 s\_2("s2");

s\_2(f1, rep, r1, r2, clk);

stage1 S\_1("stage1"); //instance of `stage1' module

//Named port binding

S\_1.in1(r1);

S\_1.in2(r2);

S\_1.sum(sum);

S\_1.diff(diff);

S\_1.clk(clk);

stage2 S\_2("stage2"); //instance of `stage2' module

S\_2(sum, diff, prod, quot, clk); //Positional port binding

stage3 S\_3("stage3"); //instance of `stage3' module

S\_3(prod, quot, powr, clk); //Positional port binding

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

D(powr, clk); //Positional port binding

sc\_initialize(); //Initialize simulation

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

clk.write(1);

sc\_cycle(10 NS);

clk.write(0);

sc\_cycle(10 NS);

}

return 0;

}

#include "systemc.h"

#include "stage1.h"

#include "stage2.h"

#include "stage3.h"

#include "S1.h"

#include "S2.h"

#include "display.h"

#include "numgen.h"

#define NS \* 1e-9

int sc\_main(int, char \*[]) {

//Signals

sc\_signal<double> in1;

sc\_signal<double> in2;

sc\_signal<double> sum;

sc\_signal<double> diff;

sc\_signal<double> prod;

sc\_signal<double> f1;

sc\_signal<double> r1;

sc\_signal<double> r2;

sc\_signal<double> rep;

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

S1 s\_1("s1");

s\_1(in1, in2, powr, f1, clk);

S2 s\_2("s2");

s\_2(f1, rep, r1, r2, clk);

stage1 S\_1("stage1"); //instance of `stage1' module

//Named port binding

S\_1.in1(r1);

S\_1.in2(r2);

S\_1.sum(sum);

S\_1.diff(diff);

S\_1.clk(clk);

stage2 S\_2("stage2"); //instance of `stage2' module

S\_2(sum, diff, prod, quot, clk); //Positional port binding

stage3 S\_3("stage3"); //instance of `stage3' module

S\_3(prod, quot, powr, clk); //Positional port binding

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

D(powr, clk); //Positional port binding

sc\_initialize(); //Initialize simulation

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

clk.write(1);

sc\_cycle(10 NS);

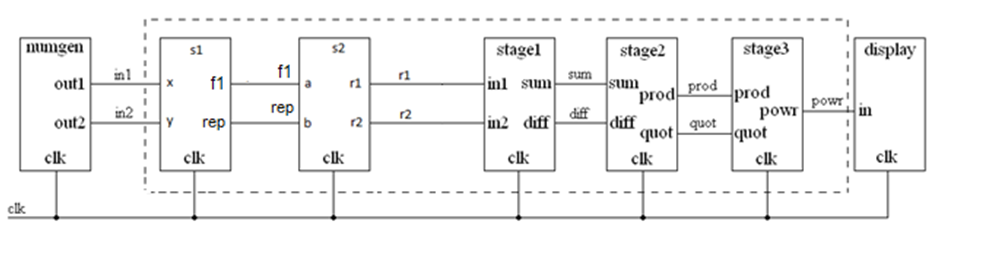
clk.write(0);

sc\_cycle(10 NS);

}

return 0;

}



*Рис.1. Структурна схема отриманої модифікованої програмної моделі*

Висновок : я навчився здійснювати аналіз програмних моделей комп’ютерних систем, виконаних на мові System C та здійснювати модернізацію функцій або параметрів арифметичного конвеєра шляхом під’єднання розроблених модулів.