**Міністерство освіти і науки України**

**Національний університет "Львівська політехніка"**

****

**Звіт до лабораторної роботи №3**

**з дисципліни “Комп’ютерні системи”**

**Аналіз програмної моделі процесу роботи арифметичного конвеєра**

Виконав ст. гр. КІ-33:

Теслюк Н.А

Прийняв: Козак Н.Б

**Львів 2020 р.**

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

**Теоретичні відомості**

Удосконалення елементної бази вже не дає кардинального росту продуктивності обчислювальної системи. Більш перспективними у цьому плані розглядаються архітектурні рішення, серед яких одне із найбільш значимих – конвеєризація.

Для пояснення ідеї конвеєра розглянемо малюнок 1, де показаний окремий функціональний блок (ФБ). Вихідні дані завантажуються у у вхідний регістр Ргвх , обробляється у функціональному блоці, а результат обробки фіксується у вихідному регістрі Ргвих. Якщо максимальний час обробки у ФБ дорівнює *Тmax ,* то нові дані можуть бути занесені у вхідний регістр Ргвх не раніше, ніж через *Тmax.*

Рг *вх* -------→ ФБ -------------------→ Рг *вих*

Рис 1. Обробка інформації у одиночному блоці.

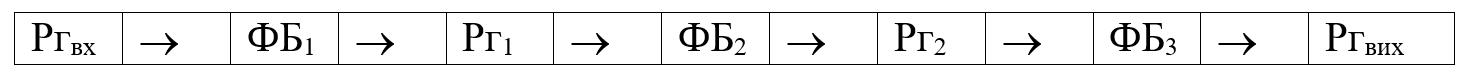


Рис. 2. Обробка інформації у конвеєрі з регістрами.

Тепер розподілемо функції, що виконуються у функціональному блоці ФБ на малюнку 1 між трьома послідовними незалежними блоками: ФБ1 , ФБ2 і ФБ3, причому так, що б максимальний час обробки у кожному ФБi був однаковий і дорівнював *Tmax/3*. Між блоками розмістимо буферні регістри Ргі, що призначені для збереження результату обробки у ФБі, на випадок, якщо наступний за ним функціональний блок ще не готовий використовувати цей результат.

Хід роботи:

**Завдання до лабораторної роботи**

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.

У моделі арифметичного конвеєра є такі інформаційні потоки:

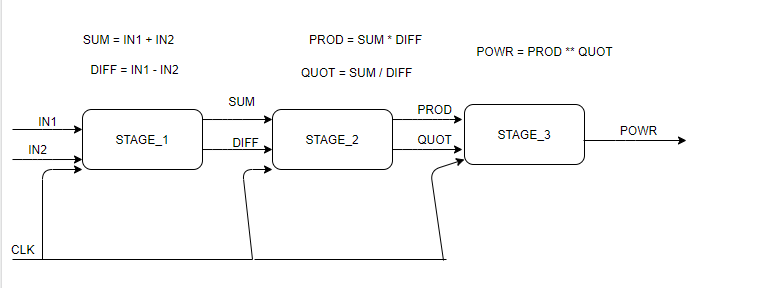
1. Спільний сигнал синхронізації clk для всіх блоків
2. Перший (вхідний) блок має 2 вхідні порти типу double, 2 вихідні порти типу double. Результат роботи першого блоку сума і різниця вхідних операндів.
3. Другий блок має 2 вхідні порти типу double. 1-ий вихідний порт видає результат множення вхідних параметрів, 2-ий — ділення.
4. Третій блок видає кінцевий результат роботи конвеєра, а саме піднесення 1-го операнда до степеня 2-го. 

Рис. 3 Схема конвеєра

**Код програми**

#include "stdafx.h"

SC\_MODULE(stage\_1) {

sc\_in<bool> clk;

sc\_in<double> in1;

sc\_in<double> in2;

sc\_out<double> sum;

sc\_out<double> diff;

void addsub() {

double a, b;

a = in1.read();

b = in2.read();

sum.write(a + b);

diff.write(a - b);

};

SC\_CTOR(stage\_1) {

SC\_METHOD(addsub);

sensitive << clk.pos();

}

};

SC\_MODULE(stage\_2) {

sc\_in<bool> clk;

sc\_in<double> sum;

sc\_in<double> diff;

sc\_out<double> prod;

sc\_out<double> quot;

void multdiv() {

double a, b;

a = sum.read();

b = diff.read();

if (b == 0) {

b = 5.0;

}

prod.write(a \* b);

quot.write(a / b);

}

SC\_CTOR(stage\_2) {

SC\_METHOD(multdiv);

sensitive << clk.pos();

}

};

SC\_MODULE(stage\_3) {

sc\_in<bool> clk;

sc\_in<double> prod;

sc\_in<double> quot;

sc\_out<double> powr;

void power() {

double a;

double b;

double c;

a = prod.read();

b = quot.read();

c = pow(a, b);

powr.write(c);

}

SC\_CTOR(stage\_3) {

SC\_METHOD(power);

sensitive << clk.pos();

}

};

SC\_MODULE(numgen) {

sc\_in<bool> clk;

sc\_out<double> out1;

sc\_out<double> out2;

void generate() {

static double a = 134.56;

static double b = 98.24;

a -= 1.5;

b -= 2.8;

cout << "a is " << a << endl;

cout << "b is " << b << endl;

out1.write(a);

out2.write(b);

}

SC\_CTOR(numgen) {

SC\_METHOD(generate);

sensitive << clk.pos();

}

};

SC\_MODULE(display) {

sc\_in<bool> clk;

sc\_in<double> sum;

sc\_in<double> diff;

sc\_in<double> prod;

sc\_in<double> quot;

sc\_in<double> powr;

void print() {

cout << "clk is " << clk << endl;

cout << "sum is " << sum << endl;

cout << "diff is " << diff << endl;

cout << "prod is " << prod << endl;

cout << "quot is " << quot << endl;

cout << "powr is " << powr << endl;

}

SC\_CTOR(display) {

SC\_METHOD(print);

sensitive << clk.pos();

}

};

int sc\_main(int argc, char\*\* argv) {

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;

clk = true;

numgen N("numgen");

N(clk, in1, in2);

stage\_1 S1("stage1");

S1(clk, in1, in2, sum, diff);

stage\_2 S2("stage2");

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

stage\_3 S3("stage3");

S3(clk, prod, quot, powr);

display D("display");

D(clk, sum, diff, prod, quot, powr);

sc\_initialize();

for (int i = 0; i < 250; i+=25) {

cout << "Time is now: " << sc\_time\_stamp() << endl;

clk.write(1);

sc\_start(25, SC\_NS);

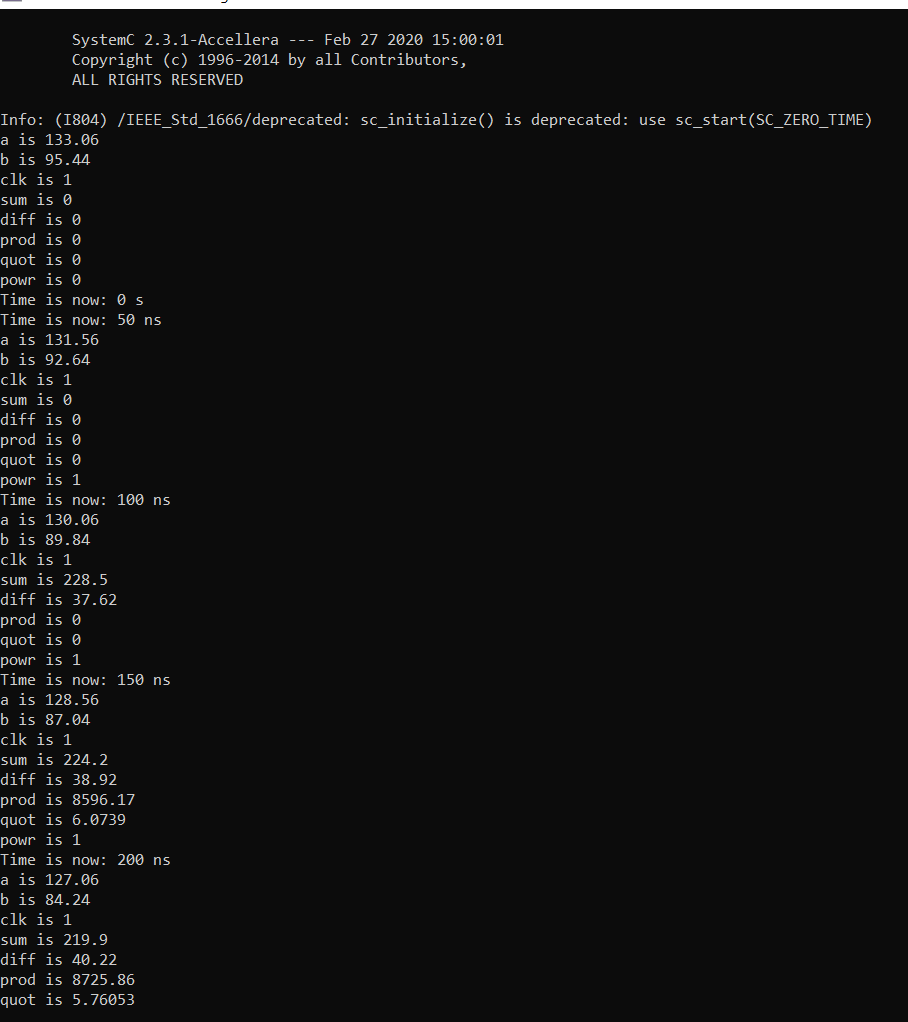
clk.write(0);

sc\_start(25, SC\_NS);

}

return 0;

}



**Висновок**

На даній лабораторній роботі я проаналізував роботу арифметичного конвеєра.