Національний технічний університет України

«Київський політехнічний інститут»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

**Лабораторна робота №7**

*з предмету: «ТПКС»*

*по темі:*

«Автоматизація генерації аналітичних форм булевих функцій з табличної форми»

Виконав: студент ФІОТ

групи ІО-92

Петрук В.О.

Київ 2012р.

**Мета:** Здобуття навичок автоматизації перетворення представлення булевих функцій з табличної до аналітичної форми для заданого елементного базису.

**Завдання**

1. Представити номер залікової книжки в двійковому вигляді:

(NNNN)10=(n15n14…n2n1)2

2. В залежності від молодших розрядів номера залікової книжки визначити

елементний базис:

n3 n2 n1 Елементний базис

0 1 1 NOT,2OR,2AND

3. Розробити модуль генерації аналітичної форми мінімізованих булевих функцій з

попередньої роботи (Лаб.6).

4. Реалізувати засоби збереження результатів у файл формату VHDL.

**Опис програми**

Для побудови на заданому базисі(NOT,2OR,2AND ) використовувалися результати мінімізації (ДКНФ). Обрана форма І-НЕ / І.

Клас And3 є програмною симуляцією елемента. Він містить дані про сигнали, що подаються на вхід, імені вихідного сигналу, прапор, що вказує на необхідність інвертора на виході.

Перед створенням VHDL файлу створюється об'єкт ксласу PreparatorForAnd3Form, в якому відбувається перетворення вхідних даних (каскадування).

VHDLcreator записує результати каскадування в файл вказаного формату.

**Код програми**

Представлені лише класи та методи, які безпосередньо були реалізовані або змінені для виконання даної лабораторної роботи.

package redactor.vhdl;

public class And3 {

private String name;

private String[] input;

private String output;

private boolean outputInverted = false;

public And3(){

input = new String[3];

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public void setConcreteInput(int index, String value){

input[index] = value;

}

public boolean isOutputInverted() {

return outputInverted;

}

public void setOutputInverted(boolean outputInverted) {

this.outputInverted = outputInverted;

}

public String getOutput() {

return output;

}

public void setOutput(String output) {

this.output = output;

}

public int returnFreeInput(){

if (input[0] == null)

return 0;

if (input[1] == null)

return 1;

if (input[2] == null)

return 2;

return -1;

}

public String getConcreteInput(int index){

return input[index];

}

}

package redactor.vhdl;

import redactor.gui.graph.mimimisation.KavaiForOneFunction;

import java.io.FileOutputStream;

import java.io.IOException;

import java.io.PrintWriter;

import java.util.List;

public class VHDLcreator {

List<KavaiForOneFunction> listOfMimimi;

String[] header;

PreparatorForAnd3Form preparatorForAnd3Form;

public VHDLcreator(List<KavaiForOneFunction> listOfMimimi, String[] header) {

this.listOfMimimi = listOfMimimi;

this.header = header;

preparatorForAnd3Form = new PreparatorForAnd3Form(listOfMimimi, header);

}

public void writeToFile(final String path) throws IOException {

preparatorForAnd3Form.prepareNodes();

List<And3> listOfNodes = preparatorForAnd3Form.getListOfNodes();

PrintWriter out = new PrintWriter(new FileOutputStream(path));

out.println("entity MyEntity is\n" +

"\tport (");

for (int i = 4; i < header.length; i++) {

String current = header[i];

if (current.contains("T")) {

out.println("\t\t" + current + " : out bit;");

out.print("\t\tQ" + current.substring(1) + " : in bit");

if (i < header.length - 1)

out.print(";");

out.println();

} else {

if (current.contains("x") || current.contains("X")) {

out.println("\t\t" + current + " : in bit;");

}

else out.println("\t\t" + current + " : out bit;");

}

}

out.println("\t);\n" + "end entity MyEntity;\n");

out.println("");

out.println("architecture MyArchitecture of MyEntity is");

out.print("\tsignal ");

for (int i = 0; i< listOfNodes.size()-1; i++){

String temp = listOfNodes.get(i).getOutput();

if (temp.contains("temp")) {

out.print(temp);

if (i< listOfNodes.size()-2)

out.print(",");

}

}

out.println(" : bit;");

out.println("begin");

for (int i = 0; i< listOfNodes.size(); i++){

And3 current = listOfNodes.get(i);

out.print("\t" + current.getName() + ": process(");

for (int k = 0; k<3; k++){

String temp = current.getConcreteInput(k);

if (temp != null){

if (temp.contains("!"))

out.print(temp.substring(1));

else out.print(temp);

if ((k<2) && (current.getConcreteInput(k+1) != null))

out.print(",");

}

}

out.println(")");

out.println("\tbegin");

out.print("\t\t" + current.getOutput() + " <= ");

String sequence = "";

if (current.isOutputInverted())

sequence += "not ";

sequence+="(";

for (int k = 0; k< 3; k++){

String temp = current.getConcreteInput(k);

if (temp != null){

if (temp.contains("!")) {

sequence += "not ";

sequence += temp.substring(1)+" ";

}

else

sequence += temp+" ";

if ((k<2) && (current.getConcreteInput(k+1) != null))

sequence += "and " ;

else sequence += ");" ;

}

}

out.println(sequence);

out.println("\tend process " + current.getName() + ";");

out.println("");

}

out.println("end architecture MyArchitecture;") ;

out.flush();

out.close();

System.out.println("write vhdl - OK");

}

}

package redactor.vhdl;

import redactor.gui.graph.mimimisation.KavaiForOneFunction;

import java.util.ArrayList;

import java.util.Iterator;

import java.util.List;

public class PreparatorForAnd3Form {

List<KavaiForOneFunction> listOfMimimi;

String[] header;

List<And3> listOfNodes;

int tempCounter;

public PreparatorForAnd3Form(List<KavaiForOneFunction> listOfMimimi, String[] header) {

this.listOfMimimi = listOfMimimi;

this.header = header;

listOfNodes = new ArrayList<And3>();

tempCounter = 0;

}

public void prepareNodes() {

for (int i = 0; i < listOfMimimi.size(); i++) {

KavaiForOneFunction currentFunction = listOfMimimi.get(i);

String[] currentHeader = currentFunction.getHeader();

List<And3> listForCurrentFunction = new ArrayList<And3>();

for (int j = 0; j < currentFunction.getMimimisated().size(); j++) {

int[] currentBraces = currentFunction.getMimimisated().get(j);

ArrayList<And3> listForCurrentBrace = new ArrayList<And3>();

And3 and3 = new And3();

and3.setOutput("temp\_" + tempCounter);

and3.setName("P\_"+tempCounter);

tempCounter++;

for (int k = 0; k < currentBraces.length-1; k++) { //because last elementis only a marker

if (currentBraces[k] != -1) {

int index = and3.returnFreeInput();

if (index != -1) {

switch (currentBraces[k]) {

case 1:

and3.setConcreteInput(index, currentHeader[k]);

break;

case 0:

and3.setConcreteInput(index, "!" + currentHeader[k]);

break;

}

} else {

listForCurrentBrace.add(and3);

and3 = new And3();

and3.setOutput("temp\_" + tempCounter);

and3.setName("P\_"+tempCounter);

tempCounter++;

switch (currentBraces[k]) {

case 1: //00000000000000000

and3.setConcreteInput(0, currentHeader[k]);

break;

case 0:

and3.setConcreteInput(0, "!" + currentHeader[k]);

break;

}

}

}

}

listForCurrentBrace.add(and3);

compositeAndObjects(listForCurrentBrace);

listForCurrentBrace.get(listForCurrentBrace.size()-1).setOutputInverted(true);

listForCurrentFunction.addAll(listForCurrentBrace);

}

List<And3> inverted = selectInverted(listForCurrentFunction) ;

compositeAndObjects(inverted);

inverted.removeAll(selectInverted(listForCurrentFunction));

inverted.get(inverted.size()-1).setOutput(currentFunction.getName());

listForCurrentFunction.addAll(inverted);

listOfNodes.addAll(listForCurrentFunction);

}

}

public List<And3> selectInverted(List<And3> list){

ArrayList<And3> inverted = new ArrayList<And3>();

for (int i = 0; i< list.size(); i++){

if (list.get(i).isOutputInverted())

inverted.add(list.get(i));

}

return inverted;

}

public void compositeAndObjects(List<And3> list) {

ArrayList<And3> temp2 = new ArrayList<And3>();

temp2.addAll(list);

ArrayList<And3> temp1 = new ArrayList<And3>();

while (temp2.size() > 1) {

temp1.addAll(temp2);

temp2 = new ArrayList<And3>();

And3 tempAndContainer = new And3();

int c = 0;

tempAndContainer.setOutput("temp\_" + tempCounter);

tempAndContainer.setName("P\_"+tempCounter);

tempCounter++;

for (Iterator<And3> i = temp1.iterator(); i.hasNext(); ) {

And3 andString = i.next();

if (c > 2){

temp2.add(tempAndContainer);

tempAndContainer = new And3();

tempAndContainer.setOutput("temp\_" + tempCounter);

tempAndContainer.setName("P\_"+tempCounter);

tempCounter++;

c = 0;

}

if (andString.returnFreeInput() == 1){

tempAndContainer.setConcreteInput(c,andString.getConcreteInput(0));

list.remove(andString);

}

else

tempAndContainer.setConcreteInput(c, andString.getOutput());

c++;

i.remove();

}

temp2.add(tempAndContainer);

list.addAll(temp2);

}

}

public void printInfo(){

for (int i = 0; i< listOfNodes.size(); i++){

And3 current = listOfNodes.get(i);

System.out.print("node " + current.getName() +": output: "+current.getOutput()+ "; input: " + current.getConcreteInput(0)

+ " " + current.getConcreteInput(1) + " " + current.getConcreteInput(2) + "; inversion ");

if (current.isOutputInverted())

System.out.println("on");

else System.out.println("off");

if (!current.getOutput().contains("temp"))

System.out.println("---------------------");

}

}

public List<And3> getListOfNodes() {

return listOfNodes;

}

}

**Висновок**

Розроблено та реалзовано модуль генерації аналітичної форм мінімізованих булевих функцій,

результатом роботи якого є VHD-файл. Для оптимізації елемент І не може мати єдиний використаний вхід (крім випадків, коли вся функція зводиться до єдиного значенням сигналу). Можливий перегляд каскадувати функції.