Project Report

A simple algorithm for K-map minimization

CSCE2301: Digital Design I

Ibrahim Gohar
Abdelaaty Rehab

The American University in Cairo

Project Report

A simple algorithm for K-map minimization

by

Ibrahim Gohar
Abdelaaty Rehab

Student Name	Student ID Number
Ibrahim Gohar	900203321
Abdelaaty Rehab	900204245

Instructor:	Dr. Mona Farouk
Teaching Assistant:	I. Sarah Taha
Project Publication Date:	Sunday 27th March, 2022
Faculty:	Department of Computer Science and Engineering ,AUC

Preface

A preface...

Ibrahim Gohar
Abdelaaty Rehab
AUC, March 2022

This is a report that reflects the process gone through developing a C++ program used for K-map minimization. This topic is crucially important because they are used to simplify most logic circuits that most functions depend on which cannot be done by human tracing truth tables only. This could be done using the advantages K-maps offer to human beings such as being simpler and less error-prone compared to the method of solving the logical expressions using Boolean laws. Moreover , it prevents the need to remember each and every Boolean algebraic theorem. It also involves fewer steps than the algebraic minimization technique to arrive at a simplified expression. K-map simplification technique always results in minimum expression if carried out properly. This is why this topic is truly important and should be mastered precisely in the region of designing digital circuits.

Summary

This report offers a short and a brief introduction to an algorithm processed to solve 3 variable K-maps. At the beginning, the user would find some instruction on how to compile, build and run the program. The user will also find a description of the program requirements, the way it receives inputs and the expected form of the output so that the user could get familiar with the running project. The main body of the report goes through the implementation of the program including a detailed explanation for the algorithm step by step and reporting the problem the user might encounter during run-time. Finally, the report appendixes the source code along with the workload distribution among the team to be reported for supervisors.

Contents

1

How to use the program

Like any other C++ based program, the program can be built and runned using minGW as the main compiler for C++ based program. Usually all Linux distibutions come with a version of minGW installed with the systems. On the other hand, if you are using any version of Windows and you do not have minGw, you can follow the installation using the following For further references click the following **link** or go to the next url: `https://www.mingw-w64.org`.
Having installed the compiler successfully, you can now run and build the program . Just use the following command on any Linux distribution

```
g++ cpp_name.cpp
./a.out
```

Listing 1.1: How to compile on Linux

or this one on any Windows version

```
g++ cpp_name.cpp
./a.exe
```

Listing 1.2: How to compile on Windows

Having compiled and opened the resulting file, you can follow the following convention in entering the inputs to the program as follows:

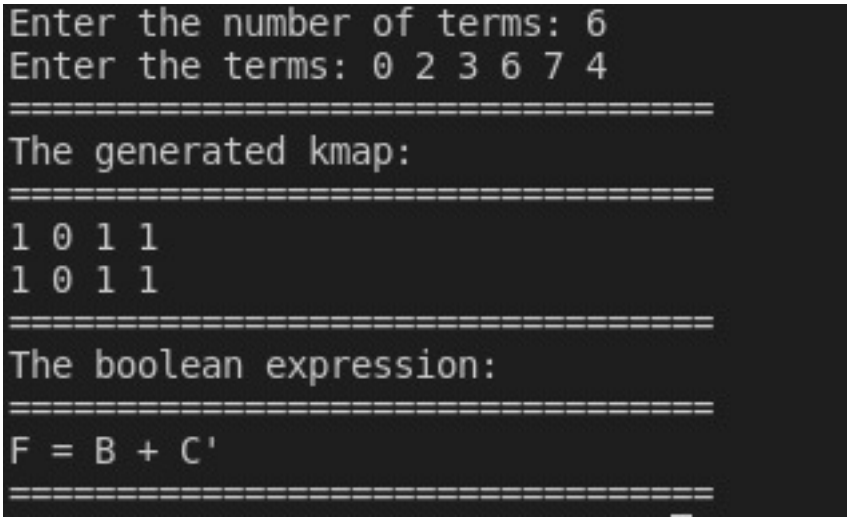


Figure 1.1: Preview of the template

2

About the Program

This program achieves three main functions which are:

- **Input:** Read in (and validate) a Boolean function using its minterms as decimal numbers.
- **K-map Generation:** Generate and print the corresponding K-map.
- **Minimization:** Generate and print the simplified Boolean expression.

2.1. Input Validation

The program reads in the input from the user and validates this input according to the functionalities of the program as follows:

- **Number of minterms validation** The user is asked to input the number of minterms he is going to use. If for any reason this input is invalid, the program will not continue unless the user provides a valid one.
- **Minterms indices verification** Since the program is designed to simplify 3 variable K-maps, thus the indices of the allowed minters are between 0 and 7 inclusive. This means that any input out of this range is invalid and the user is asked to provide a valid input instead.

2.2. Printing the K-map

The program uses the input minterms to construct the corresponding K-map. This K-map is printed using a built- in function inside the program that is designed to achieve this purpose. If a cell in the K-map contains 1, this means that this minterm was inputted by the user as true. Otherwise, it would be false of value 0. The following chapter of the report discusses the program design and the algorithm developed throughout the process.

3

Program Design

For the first glance, we thought we could use one of the available algorithms for solving this problem. However, we made a meeting with our brains to see whether we could develop another systematic algorithm to solve this problem or now and we came up with a somehow Divide and Conquer algorithm. The program tries to find the minimal number of prime implicants horizontally then vertically. The final solution of the K-map would be the result of the two answered merged together.

3.1. Horizontal Search for the largest grouping of consecutive ones

As clear from the title, the first step of the algorithm is searching for the largest grouping of ones horizontally. This means that the program finds all the prime implicants that are covering elements within the same row. These implicants are stored inside a data structure to be used later.

3.2. Vertical Search for the largest grouping of consecutive ones

The second step of the algorithm is the first one but in another dimension. The program searches for the largest grouping of ones vertically. This means that the program finds all the prime implicants that are covering elements within the same column(s). These implicants are stored inside a data structure to be used later.

3.3. Translating the output to readable Boolean expressions

The output of the first two stages is coded in numbers in the modulo 4 number system. This means that those implicants (represented in such system) have to be decoded from those meaningless numbers to some understandable Boolean expressions. This is exactly the functionality of this step. It translates the coded implicants to expressions that are stored within the program for later use.

3.4. Mapping the implicants to their corresponding cells in the K-maps

The previous steps yield some Boolean expressions whose places on the K-map are not known. Here comes the turn of the current step where each cell in the K-map is assigned the prime implicant that covers it. It could be the case that multiple implicants cover common cells, that is why each cell is assigned a vector of implicants to allow storing multiple implicants covering the same cell.

3.5. Maximization Problem

In this step, the program maximizes the size of the implicant for each cell. In other words, the program loops over all cells and chooses the largest prime implicant (in area) that covers this cell. For all cells, we guarantee that we get the minimum number of implicants because we cover the largest possible area in each step. This is done by allowing the program to compare and sort the string implicants by their areas and choosing those with maximum area for the function output.

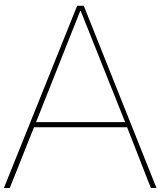
3.6. Printing the Output

This is the last stage of the program where the program takes the implicants resulting from the previous step and connect them using dis-junction to represent the final solution of the K-map.

4

Problems

The program was tested with multiple cases without showing errors in the final solution of the Karnaugh maps.



Source Code Example

Adding source code to your report/thesis is supported with the package `listings`*. An example can be found below. Files can be added using* `\lstinputlisting[language=<language>]{<filename>}`*.*

```cpp
#include <bits/stdc++.h>
using namespace std;
#define V vector
#define mat V<V<int>>
#include <unordered_set>

struct implicant {
    string name;
    int size;
    bool isEssential;
    implicant(string n, int s) {
        this->name = n;
        this->size = s;
        isEssential = false;
    }
    implicant() {
        this->name = "";
        this->size = 0;
        isEssential = false;
    }
};

bool compareByLength(const implicant& a, const implicant& b) {

    return (a.size > b.size);
}

void printMap(mat& maap) {
    for (int i = 0; i < 2; i++) {
        for (int j = 0; j < 4; j++)
            cout << maap[i][j] << " ";
        cout << "\n";
    }
}

set<pair<int, int>> inters(set<pair<int, int>>& count_1, set<pair<int, int>>& count_2) {
    set<pair<int, int>> ans;
    for (auto u : count_1) if (count_2.count(u)) ans.emplace(u);
    return ans;
}

set<pair<int, int>> convert(string implic) {
    set<pair<int, int>> ans;
    for (int i = 0; i < 2; i++) for (int j = 0; j < 4; j++) ans.emplace(i, j);
    reverse(implic.begin(), implic.end());
    while (!implic.empty()) {
        set<pair<int, int>> temp;
```

```
        char c = implic.back(); implic.pop_back();
        bool is_bar = (!implic.empty() && implic.back() == '\'');
        if (is_bar) implic.pop_back();
        if (c == 'A')
            for (int j = 0; j < 4; j++) temp.emplace(!is_bar, j);
        else {
            pair<int, int> corner = { 0, 0 };
            corner.second = (6 - (is_bar ? 2 : 0) + 'B' - c) % 4;
            temp.emplace(corner);
            temp.emplace(corner.first, (corner.second + 1) % 4);
            temp.emplace(corner.first + 1, (corner.second + 1) % 4);
            temp.emplace(corner.first + 1, corner.second);
        }
        ans = inters(ans, temp);
    }
    return ans;
}

int gfh(int idx) {
    if (idx == 2) return 3;
    return (idx == 3 ? 2 : idx);
}

void minimize_permutation(int& running_min, vector<string>& ans_, vector<string>& cur) {
    vector<string> tempo;
    vector<vector<bool>> vis(2, vector<bool>(4));
    int count = 0;
    for (auto& implic : cur) {
        set<pair<int, int>> pairs = convert(implic);
        bool is_all_vis = true;
        for (auto p : pairs) if (!vis[p.first][p.second]) {
            is_all_vis = false;
            vis[p.first][p.second] = 1;
        }
        if (is_all_vis) continue;
        tempo.emplace_back(implic);
        count++;
    }
    if (count < running_min) {
        ans_ = tempo;
        running_min = count;
    }
}

int gfh2(int idx) {
    if (idx == 6) return 7;
    return (idx == 7 ? 6 : idx);
}
int main() {
    kmap:
        vector<int> minterms;
        imp:
        cout << "Enter the Number of Terms: ";
        int n; cin >> n;
        if (n < 0 || n > 7)
        {
            cout << "==============================" << endl;
            goto imp;
        }
        int m = 0;
        cout << "Enter the Terms: ";
        while (n > 0) {
            term:
            cin >> m;
            if(m < 0 || m > 7)
            {
                cout << "Invalid." << endl;
                goto term;
            }
            minterms.push_back(m);
            n--;
```

```
        }
        if ((int)minterms.size() == 8) {
            cout << 1 << "\n";
        }
        mat maap(2, V<int>(5, 0));
        for (int i = 0; i < minterms.size(); i++) {
            int row = minterms[i] / 4;
            int col = minterms[i] % 4;
            if (col == 3)
                col = 2;
            else if (col == 2)
                col = 3;
            maap[row][col] = 1;
        }
        vector<implicant> cells[8];

        cout << "==============================" << endl;
        cout << "The Generated kmap: " << endl;
        cout << "==============================" << endl;
        printMap(maap);
        cout << "==============================" << endl;

        //horizontal search
        vector<string> imp;

        for (int i = 0; i < 2; i++) {
            string ins;
            if (i == 0)
                ins += "0";
            else
                ins += "1";
            for (int j = 0; j < 5; j++) {
                if (maap[i][j] == 1)
                    ins += to_string(gfh(j));
                else if (j != 0 && maap[i][j - 1] == 1) {
                    int end = 1 << (j / 2);
                    string one, two;
                    if (ins.size() == 4) {
                        one = ins.substr(0, 3);
                        two = ins.substr(0, 1);
                        two += ins.substr(2, 2);
                        imp.push_back(one);
                        imp.push_back(two);
                    }
                    else {
                        imp.push_back(ins);
                    }

                    ins.clear();
                    if (j != 5)
                        ins += to_string(i);
                }
            }
            if (maap[i][0] == maap[i][3] && maap[i][0] == 1 && (maap[i][2] != 1 || maap[i][1]
    != 1))
                imp.push_back(to_string(i) + "02");
        }
        vector<implicant> lett;
        map<string, implicant> hmap;
        for (int i = 0; i < imp.size(); i++) {
            if (imp[i].size() == 5) {
                if (imp[i][0] == '0') {
                    string name = "A'";
                    lett.push_back(implicant(name, 4));
                    hmap[imp[i]] = (implicant(name, 4));
                }
                else {
                    string name = "A";
                    lett.push_back(implicant(name, 4));
                    hmap[imp[i]] = implicant(name, 4);
                }
```

```
            }
            else if (imp[i].size() == 3) {
                if (imp[i][0] == '0') {
                    string name = "A'";
                    if ((imp[i][1] == '0' && imp[i][2] == '1') || (imp[i][2] == '0' && imp[i
    ][2] == '1')) {
                        name += "B'";
                    }
                    if ((imp[i][1] == '1' && imp[i][2] == '2') || (imp[i][2] == '1' && imp[i
    ][1] == '2') || (imp[i][1] == '1' && imp[i][2] == '3') || (imp[i][2] == '1' && imp[i][1]
    == '3')) {
                        name += "C";
                    }
                    if ((imp[i][1] == '2' && imp[i][2] == '3') || (imp[i][2] == '2' && imp[i
    ][1] == '3')) {
                        name += "B";
                    }
                    if ((imp[i][1] == '0' && imp[i][2] == '2') || (imp[i][2] == '0' && imp[i
    ][1] == '2')) {
                        name += "C'";
                    }
                    lett.push_back(implicant(name, 2));
                    hmap[imp[i]] = implicant(name, 2);
                }

                else if (imp[i][0] == '1') {
                    string name = "A";
                    if ((imp[i][1] == '0' && imp[i][2] == '1') || (imp[i][2] == '0' && imp[i
    ][1] == '1')) {
                        name += "B'";
                    }
                    if ((imp[i][1] == '1' && imp[i][2] == '3') || (imp[i][2] == '1' && imp[i
    ][1] == '3')) {
                        name += "C";
                    }
                    if ((imp[i][1] == '2' && imp[i][2] == '3') || (imp[i][2] == '2' && imp[i
    ][1] == '3')) {
                        name += "B";
                    }
                    if ((imp[i][1] == '0' && imp[i][2] == '2') || (imp[i][2] == '0' && imp[i
    ][1] == '2')) {
                        name += "C'";
                    }
                    lett.push_back(implicant(name, 2));
                    hmap[imp[i]] = implicant(name, 2);
                }
            }

            else if (imp[i].size() == 2) {
                if (imp[i][0] == '0') {
                    string name = "A'";
                    if (imp[i][1] == '0')
                        name += "B'C'";
                    else if (imp[i][1] == '1')
                        name += "B'C";
                    else if (imp[i][1] == '3')
                        name += "BC";
                    else if (imp[i][1] == '2')
                        name += "BC'";
                    if (maap[1][gfh(imp[i][1] - '0')] == 0) {
                        lett.push_back(implicant(name, 1));
                        hmap[imp[i]] = implicant(name, 1);
                    }
                }
                else if (imp[i][0] == '1') {
                    string name = "A";
                    if (imp[i][1] == '0')
                        name += "B'C'";
                    else if (imp[i][1] == '1')
                        name += "B'C";
                    else if (imp[i][1] == '3')
```

```
                        name += "BC";
                    else if (imp[i][1] == '2')
                        name += "BC'";
                    if (maap[0][gfh(imp[i][1] - '0')] == 0) {
                        lett.push_back(implicant(name, 1));
                        hmap[imp[i]] = implicant(name, 1);
                    }
                }

            }
        }

        for (int i = 0; i < imp.size(); i++) {
            for (int j = 1; j < imp[i].size(); j++) {
                int r = 4 * (imp[i][0] - '0');
                r += (imp[i][j] - '0');
                implicant tmp = hmap[imp[i]];
                if (tmp.name != "")
                    cells[r].push_back(hmap[imp[i]]);

            }
        }
        vector<implicant> tmp = cells[2];
        cells[2] = cells[3];
        cells[3] = tmp;
        tmp = cells[6];
        cells[6] = cells[7];
        cells[7] = tmp;


        //vertical search
        for (int i = 0; i < 4; i++) {
            if (maap[0][i] == 1 && maap[1][i] == 1) {
                if (i == 0) {
                    cells[0].push_back(implicant("B'C'", 2));
                    cells[i + 4].push_back(implicant("B'C'", 2));
                }
                else if (i == 1) {
                    cells[i].push_back(implicant("B'C", 2));
                    cells[i + 4].push_back(implicant("B'C", 2));
                }
                else if (i == 2) {
                    cells[2].push_back(implicant("BC", 2));
                    cells[6].push_back(implicant("BC", 2));
                }
                else if (i == 3) {
                    cells[3].push_back(implicant("BC'", 2));
                    cells[7].push_back(implicant("BC'", 2));
                }
            }
        }
        for (int i = 1; i < 5; i++) {
            int in = i % 4;
            if (maap[0][i - 1] == 1 && maap[1][i - 1] == 1 && maap[0][in] == 1 && maap[1][in]
    == 1) {
                if (i - 1 == 0) {
                    cells[0].push_back(implicant("B'", 4));
                    cells[1].push_back(implicant("B'", 4));
                    cells[4].push_back(implicant("B'", 4));
                    cells[5].push_back(implicant("B'", 4));
                }
                else if (i - 1 == 1) {
                    cells[1].push_back(implicant("C", 4));
                    cells[2].push_back(implicant("C", 4));
                    cells[5].push_back(implicant("C", 4));
                    cells[6].push_back(implicant("C", 4));
                }
                else if (i - 1 == 2) {
                    cells[3].push_back(implicant("B", 4));
                    cells[2].push_back(implicant("B", 4));
                    cells[7].push_back(implicant("B", 4));
```

```
                    cells[6].push_back(implicant("B", 4));
                }
                else if (i - 1 == 3) {
                    cells[0].push_back(implicant("C'", 4));
                    cells[4].push_back(implicant("C'", 4));
                    cells[3].push_back(implicant("C'", 4));
                    cells[7].push_back(implicant("C'", 4));
                }
            }
        }

        // computing the anwser
        cout << "The boolean expression: " << endl;
        cout << "============================" << endl;
        cout << "F = ";

        int count = 0;
        unordered_set<string> eq;
        vector<string> ess;

        for (int i = 0; i < 2; i++)
        {
            for (int j = 0; j < 4; j++)
            {
                if (maap[i][j] == 1)
                {
                    vector<implicant> im = cells[count];
                    if (im.size() == 1) {
                        ess.push_back(im[0].name);
                    }
                }
                count++;
            }
        }

        count = 0;
        map<string, int> to_size;
        for (int i = 0; i < 2; i++)
        {
            for (int j = 0; j < 4; j++)
            {
                if (maap[i][j] == 1)
                {
                    vector<implicant> im = cells[count];
                    //sort(im.begin(), im.end(), compareByLength);
                    if (!im.empty()) {
                        for (auto i : im)
                            eq.insert(i.name), to_size[i.name] = i.size;
                    }
                }
                count++;
            }
        }

        vector<string> cur;
        for (auto& implic : eq) cur.emplace_back(implic);
        vector<string> ans_ = cur;
        int running_min = cur.size();
        sort(cur.begin(), cur.end(), [&](string& str1, string& str2) { return to_size[str1] >
    to_size[str2];  });
        minimize_permutation(running_min, ans_, cur);
        cur = ans_;
        sort(cur.begin(), cur.end());
        do {
            minimize_permutation(running_min, ans_, cur);
        } while (next_permutation(cur.begin(), cur.end()));
        int ss = 0;
        bool fg = true;
        for (auto& itr : ans_) {
            if (ss == ans_.size() - 1)
            {
```

```
                cout << itr;
                fg = false;
            }
            else
            {
                cout << itr << " + ";
                fg = false;
            }
            ss++;
        }
        if (fg) cout << 0;
        cout << "\n";
        cout << "==============================" << endl;
        cout << "Again? (y or n) ";
        try {
            char b;
            cin >> b;
            if (b == 'y' or b == 'Y')
            {
                cout << "==============================" << endl;
                goto kmap;
            }
            else {
                cout << "==============================" << endl;
                cout << "Thanks." << endl;
                cout << "==============================" << endl;
                return 0;
            }
            throw b;
        }
        catch (char b) {
            cout << "==============================" << endl;
            cout << "Thanks." << endl;
            cout << "==============================" << endl;
            return 0;
        }
}
```

B

Task Division Example

Tasks were divided among the group members as follows.

Table B.1: Distribution of the workload

Task	Student Name(s)
Input and Validation	Abdelaaty
K-map Construction	Abdelaaty
Horizontal Search	Abdelaaty
Decoding Implicants	Ibrahim, Abdelaaty
Vertical Search	Ibrahim
Implicants Mapping	Ibrahim, Abdelaaty
Merging Solutions	Ibrahim
Filtering Output	Ibrahim
Document Design and Layout	Abdelaaty