CSCE2301 – Digital Design I

FALL 2020

Project 1

Quine-McCluskey Logic Minimization

Andrew Sinout Shenouda – 900182668  
Mohi El-Din Hashem – 900184260

Objective:

The objective of this project is to make you more familiar with the Quine-McCluskey Logic Minimization algorithm.

Functions used in the program:

1. bool minterms(string filename, int& var, vector<int>& minterm, vector<int>& dontcare)
2. vector<pair<string,string>> dec\_to\_bin(int var, vector<int>& minterm)
3. int count\_ones(int s,string x)
4. bool found (int x, string y)
5. vector<pair<string,string>> comparison (int v, vector<pair<string,string>> a, vector<pair<string,string>> b, vector<bool>& check\_a,vector<bool>& check\_b)
6. int main ()
7. bool minterms(string filename, int& var, vector<int>& minterm, vector<int>& dontcare):

* Inputs:

1. This is a Boolean function that takes a file name in the format of string.
2. An empty variable called var in the format of int by reference.
3. Two empty vectors named minterm and dontcare in the format of vector of int by reference too.

* Implementation:

The goal of this function is to validate the inputs from the text file before the QM process. We first, open the file inside the function and extract the information into the parameters that we passed by reference above. The first line of the text file is the number of variables which is saved in the var parameter. The second line contains minterms and we use built in functions and string streams to push the minterms in the form of integers into the minterm array. We do the exact same things with the third line which contains the don’t care terms and we push them into the dontcare vector. Now that we have extracted our data, we need to validate our data. The way we perform this validation is that we make sure that the minterms and don’t care terms are always one less than two to the power the number of variables. When this condition is satisfied, the function returns true. Otherwise, it returns false and in the main program it outputs a message saying that the data is incorrect.

* Output:

1. It returns true if there are no any errors and if there any errors regarding the number of variables and the minterms it returns false.
2. It also returns the number of variables and the vectors of the minterms and the don’t care terms by reference.
3. vector<pair<string,string>> dec\_to\_bin(int var, vector<int>& minterm):

* Inputs:

1. A variable called var in the format of int that holds the number of variables.
2. A vector of int that is passed by reference.

* Implementation:

This function loops on minterms and on every minterm loops until variable called n become zero, then using % 2, it converts the every decimal into binary and stores both the decimal and the binary in a pair of strings, then it add zeros to the binary number if its length is less than the number of variables to make all binary number will same length then at the end push the pair into a vector of pairs.

* Output:

1. The function returns vector of pairs of minterms or the don’t care terms, where the pair hold the midterms or the don’t care terms in binary and the other string stores them in decimal that corresponds to.
2. int count\_ones(int s,string x):

* Inputs:

1. A variable called s in the format of int that holds the number of variables.
2. A string that contains zeros and ones.

* Implementation:

This function loops on the string x character by character and add to a counter if the character is equal to 1, then return the counter to group all minterms and don’t care terms according to the number of ones in each of them.

* Output:

1. This function returns int value which contains the number of ones in that string.
2. bool found (int x, string y):

* Inputs:

1. A variable called s in the format of int that have the minterm that want to search for.
2. A string y that contains the minterms of combined implicant that we want to search for the minterm between them.

* Implementation:

We use this function to confirm that the int x, is in fact a minterm present in the string of minterms corresponding to the desired implicant. We use the same procedure as in the minterm function to turn the string into a vector of integers and then compare the values with x to return true if it is present, and false otherwise.

* Output:

1. This function returns true if the minterm x is found in string y.
2. vector<pair<string,string>> comparison (int v, vector<pair<string,string>> a, vector<pair<string,string>> b, vector<bool>& check\_a,vector<bool>& check\_b):

* Inputs:

1. A variable called v in the format of int that holds the number of variables.
2. Two vector of pairs that stores two groups of the implication table.
3. Two Boolean vectors that corresponds to the groups sent and sent to the function by reference.

* Implementation:

This function loops on two groups and compare each element in each group to each other to see if the difference between them one character or more. If it is one character, it combines them changing the different character into dash and also using the Boolean vectors, it indicates if the element used to obtain combined terms or not. Then it loops on the checker vectors to store them in two vectors in the right order to pass it by reference, then returns the combined group as a vector.

* Output:

1. A vector of pair after combining the two groups into one group.
2. Two Boolean vectors that corresponds to the groups sent by reference, to indicate which elements of the two groups sent are prime implicants.
3. int main ():

* Instantiated variables in the beginning of the main
* int v,s; (v is equal to the number of variables) and (s stores the number of ones of each minterm or don’t care term when needed)
* string name; (name stores the path of the input file that needed to get the data)
* vector<int> m; (m is a vector that stores the minterms as integers)
* vector<int> d; (d is a vector that stores the don’t care terms as integers)
* vector<pair<string,string>> mm; (mm is a vector of pairs that stores the minterms as strings in binary and in decimal)
* vector<pair<string,string>> dd; (dd is a vector of pairs that stores the don’t care terms as strings in binary and in decimal)
* vector<vector<pair<string,string>>> groups; (groups is a vector that stores the pairs of the minterms and the don’t care terms as groups when we divide them according to the number of ones inside them)
* vector<vector<vector<pair<string,string>>>> itterations; (This is a vector stores the columns of the implicant table as vector groups that stores the pairs of the minterms and the don’t care terms as groups)
* vector<pair<string,string>> PI; ( This is a vector of pairs that stores the prime implicants)
* bool b; (This is a Boolean variable that returns true if the file is extracted successfully and false if there is any errors)
* Implementation:

The main starts by calling the minterms function and saving it into the boolean variable b. The minterm function takes in the name of the file, the integer v to store the number of variables, a vector of a string pairs mm to store the minterms, and a vector of a string pairs dd to store the don't care values. If the value of b is true then we continue the main program regularly, however if the value of b is false then this means that the information in the text file is incorrect, and the program ends immediately while printing a message that the values are invalid. After confirming that b is true, we then convert the decimal terms into their binary forms so that we may use them in the implicant table. Using the newly obtained binary forms we are able to use 2 for loops to make groups of binary number from the minterms and don't care terms based on the number of ones in each binary number, therefore creating the first column of the implicant table. We now declare a 2d vector called groups, which stores all the vectors that represent each group. In the next part, we do not really know how many columns will exist as it differs based on the number of variables. So, we now make a 3d vector that takes in the 2d vector groups so that it stores the columns that follow and stores the groups that we did in the first column of the 3d vector. We then iterate on the number of the groups sending group I and group i+1 to the function comparison to return the combined group and mark the check\_a and check\_b vectors of whether each element is combined or not to push the ones which did not get compared to the prime implicants vector. When i=0 this means that check\_a mark the right prime implicants but if i not equal to zero that means that groupi will be compared twice, so we store check\_b it in check\_c and send it again as check\_a to update if any comparisons have taken place with the next group. We also push the last column using loop as all of the last column contains prime implicants. Then we introduce PII a vector of pairs, and we made nested loops on the prime implicants vector and it self to check if the prime implicant is repeated store it once in PII. And then we print the prime implicants and its boolean expression.

For this next part we will now move on to the coverage chart. We use he EPI vector to store the essential prime implicants and the index vector to store the . We use the following two for loops to see how many times a specific minterm is present in the prime implicants we have using the found function described earlier. If it is pressent only once then this means there is only one "cross" in that column and therefore the prime implicant that the minterm is a part of is now essential and we add it to the EPI vector.

After that we made vector called r that gets the used minterms in the essential prime implicants, then by looping on the minterms we remove the elements in r from the minterms vector. Then we generate PI\_C, storing in it the prime implicants that didn’t appear in the essential prime implicants vector. Then we looped on the PI\_C and the minterms after removing r to see if their a column that is fully filled using the function found and if yes we remove the minterm that present the dominating column, and then if there is a row that is fully filled using the function found to push it in the essential prime implicants vector and if there is no dominating rows, then add the remaining terms in PI\_C in the essential prime implicants but once for every minterm.

Test cases:

1. F(A,B,C) = ∑ m(4,5,6,8,9) 🡪 error messages that minterms 8 and 9 exceed the number of the variables then it ends the program.
2. F(A,B,C) = ∑ m(1,3,6,7) + ∑ d(0,5) 🡪

The prime implicatns :

00\_(0,1)

11\_(6,7)

\_\_1(1,5,3,7)

----------------------------

The boolean equation for the prime implicatns :

A'B'+AB+C

----------------------------

Essential prime implicants :

\_\_1(1,5,3,7)

11\_(6,7)

----------------------------

The prime implicatns after removing the essential prime implicants :

00\_(0,1)

----------------

Essential prime implicatns After the dominationg colomns table :

\_\_1(1,5,3,7)

11\_(6,7)

The boolean equation for the Essential prime implicatns :

C+AB

1. F(A,B,C,D) = ∑ m(4,5,6,8,9,10,13) + ∑ d(0,7,15) 🡪

The prime implicatns :

0\_00(0,4)

\_000(0,8)

100\_(8,9)

10\_0(8,10)

1\_01(9,13)

01\_\_(4,6,5,7)

\_1\_1(5,7,13,15)

----------------------------

The boolean equation for the prime implicatns :

A'C'D'+B'C'D'+AB'C'+AB'D'+AC'D+A'B+BD

----------------------------

Essential prime implicants :

01\_\_(4,6,5,7)

10\_0(8,10)

----------------------------

The boolean equation for the Essential prime implicatns :

A'B+AB'D'

Essential prime implicatns After the dominationg colomns table :

01\_\_(4,6,5,7)

10\_0(8,10)

----------------

Essential prime implicatns After the coverage table :

01\_\_(4,6,5,7)

10\_0(8,10)

1\_01(9,13)

----------------------------

The boolean equation for the Essential prime implicatns After the coverage table :

A'B+AB'D'+AC'D

1. F(A,B,C,D) = ∑ m(4,5,6,8,9,10,13) 🡪

The prime implicatns :

010\_(4,5)

01\_0(4,6)

100\_(8,9)

10\_0(8,10)

\_101(5,13)

1\_01(9,13)

----------------------------

The boolean equation for the prime implicatns :

A'BC'+A'BD'+AB'C'+AB'D'+BC'D+AC'D

----------------------------

Essential prime implicants :

01\_0(4,6)

10\_0(8,10)

----------------------------

The boolean equation for the Essential prime implicatns :

A'BD'+AB'D'

Essential prime implicatns After the dominationg colomns table :

01\_0(4,6)

10\_0(8,10)

----------------------------

The boolean equation for the Essential prime implicatns After the dominationg colomns

A'BD'+AB'D'

1. F(A,B,C,D) = ∑ m(0,2,5,6,7,8,10,12,13,14,15) 🡪

The prime implicatns :

\_0\_0(0,8,2,10)

\_\_10(2,10,6,14)

1\_\_0(8,12,10,14)

\_1\_1(5,13,7,15)

\_11\_(6,14,7,15)

11\_\_(12,14,13,15)

----------------------------

The boolean equation for the prime implicatns :

B'D'+CD'+AD'+BD+BC+AB

----------------------------

Essential prime implicants :

\_0\_0(0,8,2,10)

\_1\_1(5,13,7,15)

----------------------------

The boolean equation for the Essential prime implicatns :

B'D'+BD

Essential prime implicatns After the dominationg colomns table :

\_0\_0(0,8,2,10)

\_1\_1(5,13,7,15)

----------------------------

The boolean equation for the Essential prime implicatns After the dominationg colomns

B'D'+BD----------------

Essential prime implicatns After the coverage table :

\_0\_0(0,8,2,10)

\_1\_1(5,13,7,15)

\_\_10(2,10,6,14)

1\_\_0(8,12,10,14)

----------------------------

The boolean equation for the Essential prime implicatns After the coverage table :

B'D'+BD+CD'+AD'

1. Test case to 20 variables:

F(A,B,C,D…up to 20 variables) = ∑ m (33,66,77,22,65,65) 🡪

The prime implicatns :

00000000000000100001(33)

00000000000001000010(66)

00000000000001000001(65)

00000000000000010110(22)

00000000000001001101(77)

----------------------------

The boolean equation for the prime implicatns :

A'B'C'D'E'F'G'H'I'J'K'L'M'N'OP'Q'R'S'T+A'B'C'D'E'F'G'H'I'J'K'L'M'NO'P'Q'R'ST'+A'B'C'D'E'F'G'H'I'J'K'L'M'NO'P'Q'R'S'T+A'B'C'D'E'F'G'H'I'J'K'L'M'N'O'PQ'RST'+A'B'C'D'E'F'G'H'I'J'K'L'M'NO'P'QRS'T

----------------------------

Essential prime implicants :

00000000000000100001(33)

00000000000001000010(66)

00000000000001001101(77)

00000000000000010110(22)

00000000000001000001(65)

00000000000001000001(65)

----------------------------

The boolean equation for the Essential prime implicatns :

A'B'C'D'E'F'G'H'I'J'K'L'M'N'OP'Q'R'S'T+A'B'C'D'E'F'G'H'I'J'K'L'M'NO'P'Q'R'ST'+A'B'C'D'E'F'G'H'I'J'K'L'M'NO'P'QRS'T+A'B'C'D'E'F'G'H'I'J'K'L'M'N'O'PQ'RST'+A'B'C'D'E'F'G'H'I'J'K'L'M'NO'P'Q'R'S'T+A'B'C'D'E'F'G'H'I'J'K'L'M'NO'P'Q'R'S'T

Essential prime implicatns After the dominationg colomns table :

00000000000000100001(33)

00000000000001000010(66)

00000000000001001101(77)

00000000000000010110(22)

00000000000001000001(65)

00000000000001000001(65)

----------------------------

The boolean equation for the Essential prime implicatns After the dominationg colomns :

A'B'C'D'E'F'G'H'I'J'K'L'M'N'OP'Q'R'S'T+A'B'C'D'E'F'G'H'I'J'K'L'M'NO'P'Q'R'ST'+A'B'C'D'E'F'G'H'I'J'K'L'M'NO'P'QRS'T+A'B'C'D'E'F'G'H'I'J'K'L'M'N'O'PQ'RST'+A'B'C'D'E'F'G'H'I'J'K'L'M'NO'P'Q'R'S'T+A'B'C'D'E'F'G'H'I'J'K'L'M'NO'P'Q'R'S'T----------------

----------------

Essential prime implicatns After the coverage table :

00000000000000100001(33)

00000000000001000010(66)

00000000000001001101(77)

00000000000000010110(22)

00000000000001000001(65)

00000000000001000001(65)

----------------------------

The boolean equation for the Essential prime implicatns After the coverage table :

A'B'C'D'E'F'G'H'I'J'K'L'M'N'OP'Q'R'S'T+A'B'C'D'E'F'G'H'I'J'K'L'M'NO'P'Q'R'ST'+A'B'C'D'E'F'G'H'I'J'K'L'M'NO'P'QRS'T+A'B'C'D'E'F'G'H'I'J'K'L'M'N'O'PQ'RST'+A'B'C'D'E'F'G'H'I'J'K'L'M'NO'P'Q'R'S'T+A'B'C'D'E'F'G'H'I'J'K'L'M'NO'P'Q'R'S'

1. F(A,B,C,D) = ∑ m (0) + ∑ d (4,6,8,3) 🡪

The prime implicatns :

0\_00(0,4)

\_000(0,8)

01\_0(4,6)

----------------------------

The boolean equation for the prime implicatns :

A'C'D'+B'C'D'+A'BD'

----------------------------

Essential prime implicants :

(no Essential prime implicants)