HW6- Report

CIS554- Object Oriented Programming
Storing combinatorial circuits in Database

Name:	Pradhyumn Bhale
SUID:	3116245952
Primary Instructor:	Prof. Roger Chen

1 Project Summary

In this project, it is required to store the combinatorial circuits in a database. Store combinational logic circuits with one or more outputs to a database, implemented using unordered map. The design has to be realistic for applications. It needs to be noted that permuting input columns, output columns, or rows will not affect the function of the circuit. The inputs are to be read from a text file and stored to the database. The circuits need to be added one-by-one using DB[key]=value. When adding a circuit, if a duplicate is detected, the program will print a message "The circuit is already in DB."

2 Input Logic

The code is reading the input from data.txt file specified in the format described in the problem statement. The structure for reading the input file is:
$unorderedmap < vector < int >, pair < vector < vector < int >>, vector < vector < int >>>, Circuit, Circuit >$
The value part of the map contains a pair of vector of vector of integers which contain the inputs and outputs of the truth table, respectively. The key part of the map is generated using a generate key function which generates a unique key for a unique truth table as described in the next section.

2.1 Key Generation

The key part of the map is a vector of int. Consider a truth table with 3 inputs and 2 outputs as shown in the figure. So, we can note the following points:

1. if the no. of inputs or the no. of outputs of the truth tables are different, then they are different.
2. if the output of inputs '000' or '111' are different, then they are different.

This means that inputs like (001), (010) and (100) and so on for inputs with 2 ones, are effectively the ones which can have a permutation amongst them and still the circuit may be the same. For this reason, the inside vector of int for both inputs and outputs are sorted as shown in the Figure. Then, these sorted input and output values are converted from their binary form to their decimal forms and multiplied row-wise. This helps in creating a unique key for each circuit. It needs to be noted that the input (000) will have a decimal form of (0) and this information can be lost in this process. To cover for this, the output of the input with all zeros is stored separately in the key part of the structure.
So, the key part consists of the no. of inputs, no. of outputs, output of input with all zeros and a unique key, which is the sum of decimal multiplications of the row-wise inputs and outputs, respectively. This key is then used for creating the hashing and equal to operator.

2.2 Hashing and Equal to:

The hashing operator uses sums the values of all the integer elements of the vector which comprises the key. If the element is even, then it is added once, and if it is odd, then it is added thrice.
The equal to operator each of the 4 elements of the key and increments a count. If the count is 4 which corresponds to each of the element being equal between the two truth tables, then, it returns true value for the two truth tables being identical.

2.3 Other functions:

There is another function binary to decimal conversion which is used in the code which converts a vector of integer which has a binary structure stored in it and returns an integer which is in the decimal form.

Sorting of the inputs and outputs in the circuit.

Inputs	Outputs	Sorted Inputs	Sorted Outputs
0 0 0	1 0	0 0 0	0 1
0 0 1	0 0	0 0 1	0 0
0 1 0	0 0	0 0 1	0 0
0 1 1	1 0	0 1 1	0 1
1 0 0	0 0	0 0 1	0 0
1 0 1	1 1	0 1 1	1 1
1 1 0	0 1	0 1 1	0 1
1 1 1	1 1	1 1 1	1 1