Assignment #1

Implement either the Quine-McCluskey 2-level minimization algorithm, or the *-operation and #-operation algorithms discussed in class using the C programming language (if you wish to use other programming languages, ask about this in lectures or through email). Your program is to take a BLIF file as input and produce an output that describes the number of cubes used in the minimum cost solution for a given function, as well as the numerical cost of the optimal 2-level representation.

To help you with this assignment, a starter kit has been provided on the course website. The starter kit contains a program to read a BLIF file and provides you a cubical function representation for use inside of the package.

BLIF File

A BLIF, or Berkeley Logic Interchange Format, file is a method of representing logic expressions as a set of cubes. A cube is represented by 0s, 1s and '-'s (hyphens). A hyphen represents an X as discussed in class. Also, a cube can be specified to cover a set of don't cares or minterms. Here is an example of how to represent logic functions in a BLIF format:

```
# BLIF format is as follows:
# start the file with a .model <circuit_name> line
.model top

# List all inputs to the circuit in the .inputs line
.inputs a b c d

# List all outputs in the .outputs line
.outputs f

# For each function in the circuit specify its inputs and outputs in the .names line
# note that the last entry in the line is the function output.
# Then list the cubes for the given function, followed by 1 to indicate a cube covering minterms
# or a - to indicate a cube of don't cares.
.names a b c d f
01-1 1
-110 1
1011 1
1-01 1
1010 -

#end the file with the .end line
.end
```

Data Representation in the package

The provided package represents a logic function as a set of cubes. This is accomplished by using two data structures:

```
typedef struct s_blif_cube {
        int data_size;
        /* Number of long ints used */

        long int signal_status[4];
        /* For each cube the signal entry can be either 0, 1, or X.
          */

        t_boolean is_DC;
        /* is cube a DC? */
} t_blif_cube;
/* This structure defines the cube used in logic function representation. */

typedef   struct s_blif_cubical_function {
        int input_count;
        /* Number of inputs to the function */

        t_blif_signal **inputs;
        /* List of input signals */

        t_blif_signal *output;
        /* Logic function output */

        int cube_count;
        /* Number of cubes */

        t_blif_cube **set_of_cubes;
        /* Set of cubes */

        int value;
        /* If there are no cubes then this stores the constant value the
        function evaluates to. Otherwise, should be < 0 */
} t_blif_cubical_function;
```

The t_blif_cube structure represents a single cube. The important field in this structure is signal_status, as it contains the set of 0's, 1's and X's for the cube in an ordered fashion. To read the state of a variable for a given cube do the following:

```
int read_cube_literal(t_blif_cube *some_cube, int literal_index)
{
        return read_cube_variable(some_cube->signal_status, literal_index);
}
```

To set the state of a specific variable in a cube do

```
int write_cube_literal(t_blif_cube *some_cube, int literal_index, int value)
{
        return write_cube_variable(some_cube->signal_status, literal_index, value);
}
```

Please note that the *value* must be either LITERAL_0, LITERAL_1, or LITERAL_DC. These constants are defined in the code for you. Also, one of the above constants will be returned to you when you read the state of a variable.

A function as a whole is represented in the t_blif_cubical_function structure. The important fields are:

- input_count - number of function inputs
- cube_count - number of cubes
- set_of_cubes - an array of cubes, represented by pointers to t_blif_cube structures.

To examine a cube in the specified function do:

```
t_blif_cube *cube_to_examine = f->set_of_cubes[index];
```

Initial Behaviour

Initially when you compile the starter kit, it should work only in its capacity to read-in a BLIF file. When provided an input file, the starter kit will read all logic functions specified in the BLIF file and then list them in order. For each function, the starter kit will provide the number of inputs it uses, the number of cubes (including don't cares) it used, as well as the cost of implementation of the function if all specified cubes are used without optimization.

Questions

The package has been tested on a Linux system at UofT, specifically on ra.eecg.utoronto.ca machine, as well as on Windows XP professional, using the Microsoft Visual Studio 7.0. If you have trouble compiling the package, or running it on a sample BLIF file provided on the website, please contact prof.brown@gmail.com or czajkow@eecg.utoronto.ca for assistance.