Switching Theory

Assignment 2

BDD Package, Apply Algorithm and Sifting Algorithm

Professor Brown

Jing Nan Chen
Student #: 1002821404

Qi Jian Huang
Student #: 1002675866

Contents

Introduction:

Assignment 2 is to implement the BDD package and use the package to perform apply and sifting algorithm on BDDs. Apply algorithm and BDD building are straight forward to develop, while Sifting algorithm took us a lot more time to figure out.

How to run our program:

Our program can be run using either the provided executable file (apply.exe or sifting.exe) or visual studio IDE (2015 version is recommended).

If by executable file:

For apply.exe:

- Use command window navigate to the directory where the executable file is located
- Make sure the test nodes also located in the same directory
- Execute “apply.exe <BLIF file> <Operator>” where BLIF file is the name of the input file and operator is the function going to apply on two functions in the BLIF. For example, enter “apply.exe node6.blif AND” to apply logical and to two BDDs. To run the apply.exe, it has to have total 3 input arguments and also the BLIF file should have exactly two functions.
- The program result should be displayed on the command window

For sifting.exe：

- Use command window navigate to the directory where the executable file is located
- Make sure the test nodes also located in the same directory
- Execute “apply.exe <BLIF file>” where BLIF file is name of the input file. For example, enter “apply.exe node6.blif” to perform sifting on the function in node6.blif.
- The program result should be displayed on the command window

If by visual studio 2015:

For Apply：

- Double click the “apply.sln”, and open our project in Visual Studio 2015
- Add test node argument and the operator under “Project -> apply properties -> debugging -> command arguments”. For example, you can type “node6.blif AND” as the command arguments.
- Run the program by clicking “Debug -> start without debugging”
- The program result should be displayed on the command window

For Sifting：

- Double click the “assign2.sln”, and open our project in Visual Studio 2015
- Add test node argument under “Project -> assign2 properties -> debugging -> command arguments”
- Run the program by clicking “Debug -> start without debugging”

- The program result should be displayed on the command window

Important features of our implementation:

- The sifting algorithm records one of the best variable ordering, which results in a smallest BDD size.
- We added level, flag, newLow and newHigh fields to a BDD node to manipulate the T table of BDD for sifting algorithm.
- When swapping two adjacent variables, we find out there would be a few cases need to be taken care of in the BDD T table. For example, if we swap x2 and x3, and x2 currently is in front of the x3 in terms of variable order. The x2 variable in the BDD could have more than one node. Therefore we have to find out all the x2 node first. Then we will have to take care of the following cases:
    - x2 node has both low and high child link to x3 node
    - x2 node has either low or high child link to x3 node

  For the above cases, we insert two x2 nodes, and for sure we need to check the node redundancy to keep the BDD reduced. Then we add a new x3 node and make its low and high child link to the two new x2 nodes. The we swap the level of x2 and x3 nodes.

  If x2 node has no link to x3 node, then we don't do anything but just swap the level information of these two nodes. If x3 has more than one parents and the

extra parent is not x2. We should keep this x3 node and do not delete it during the cleaning.

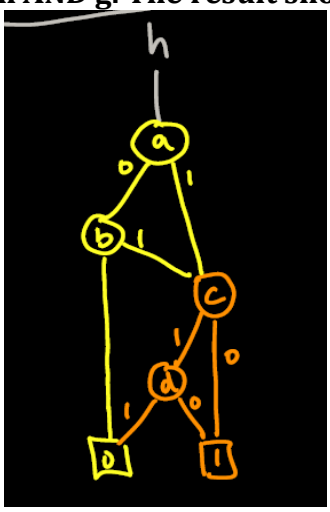
We don't delete any x3 nodes until we go through all the x2 nodes. Then we have to clean the table by removing the x3 nodes that marked as useless. Then we update the T table based on the level information.

- The sifting algorithm will swap a variable to the rightmost of the variable ordering and swap it to the leftmost of the variable ordering to find the smallest BDD
- The H table in the BDD package is a hash table structure and it is from Frank Pfenning, since we don't have enough time to implement our own hash table in C. And we believe using hash table will improve the performance of the program.

Test Results apply.exe:

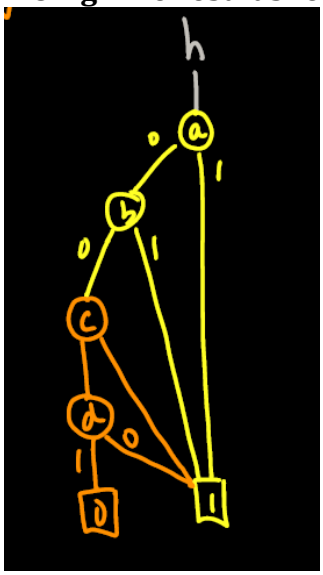
1. **Example from Lecture Slide (in Lecture slide Page 44 to Page 46 in lecture1-7.pdf).**
   h = a+ b, g = c' + cd', where in our implementation, a, b, c, d will be x1, x2, x3 and x4 respectively. The results are all correct for this test. And the runtimes are 37ms, 48ms, and 52ms.

- **h AND g. The result should be:**



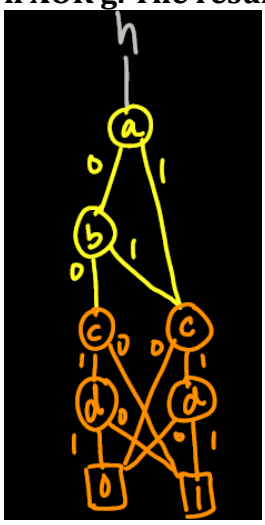

```
PS D:\Dropbox\Dropbox\_MEng\Switching\assignment2\Executable
                ECE1733 - Switching Theory - Assignment 2
                 Reduced Ordered Binary Decision Diagram.
Reading file .\lecture_node.blif...
Function 1: #inputs = 2; #cubes = 2;
Cube 0: 1-
Cube 1: -1
=============================================
!u !     var !    low !    high !   flag !   level !
!0 !       5 !        !         !     0  !     5   !
!1 !       5 !        !         !     0  !     5   !
!2 !       2 !      0 !       1 !     0  !     2   !
!3 !       1 !      2 !       1 !     0  !     1   !
=============================================
Function 2: #inputs = 2; #cubes = 2;
Cube 0: 0-
Cube 1: 10
=============================================
!u !     var !    low !    high !   flag !   level !
!0 !       5 !        !         !     0  !     5   !
!1 !       5 !        !         !     0  !     5   !
!2 !       4 !      1 !       0 !     0  !     4   !
!3 !       3 !      1 !       2 !     0  !     3   !
=============================================

APPLY RESULT:
=============================================
!u !     var !    low !    high !   flag !   level !
!0 !       5 !        !         !     0  !     5   !
!1 !       5 !        !         !     0  !     5   !
!2 !       4 !      1 !       0 !     0  !     4   !
!3 !       3 !      1 !       2 !     0  !     3   !
!4 !       2 !      0 !       3 !     0  !     2   !
!5 !       1 !      4 !       3 !     0  !     1   !
=============================================
Time for Apply = 37.000000 ms
Done.
```

- **h OR g. The result should be:**




```
PS D:\Dropbox\Dropbox\_MEng\Switching\assignment2\Executabl
                ECE1733 - Switching Theory - Assignment 2
                 Reduced Ordered Binary Decision Diagram.
Reading file .\lecture_node.blif...
Function 1: #inputs = 2; #cubes = 2;
Cube 0: 1-
Cube 1: -1
=============================================
!u !     var !    low !    high !  flag !   level !
!0 !       5 !        !         !     0  !     5   !
!1 !       5 !        !         !     0  !     5   !
!2 !       2 !      0 !       1 !     0  !     2   !
!3 !       1 !      2 !       1 !     0  !     1   !
=============================================
Function 2: #inputs = 2; #cubes = 2;
Cube 0: 0-
Cube 1: 10
=============================================
!u !     var !    low !    high !  flag !   level !
!0 !       5 !        !         !     0  !     5   !
!1 !       5 !        !         !     0  !     5   !
!2 !       4 !      1 !       0 !     0  !     4   !
!3 !       3 !      1 !       2 !     0  !     3   !
=============================================

APPLY RESULT:
=============================================
!u !     var !    low !    high !  flag !   level !
!0 !       5 !        !         !     0  !     5   !
!1 !       5 !        !         !     0  !     5   !
!2 !       4 !      1 !       0 !     0  !     4   !
!3 !       3 !      1 !       2 !     0  !     3   !
!4 !       2 !      3 !       1 !     0  !     2   !
!5 !       1 !      4 !       1 !     0  !     1   !
=============================================
Time for Apply = 48.000000 ms
Done.
```

- **h XOR g. The result should be:**




```
PS D:\Dropbox\Dropbox\_MEng\Switching\assignment2\Executables\Apply> .\app
                ECE1733 - Switching Theory - Assignment 2
                 Reduced Ordered Binary Decision Diagram.
Reading file .\lecture_node.blif...
Function 1: #inputs = 2; #cubes = 2;
Cube 0: 1-
Cube 1: -1
=============================================
|u |     var |    low |    high |  flag |  level |
|0 |       5 |        |         |     0 |     5  |
|1 |       5 |        |         |     0 |     5  |
|2 |       2 |      0 |       1 |     0 |     2  |
|3 |       1 |      2 |       1 |     0 |     1  |
=============================================
Function 2: #inputs = 2; #cubes = 2;
Cube 0: 0-
Cube 1: 10
=============================================
|u |     var |    low |    high |  flag |  level |
|0 |       5 |        |         |     0 |     5  |
|1 |       5 |        |         |     0 |     5  |
|2 |       4 |      1 |       0 |     0 |     4  |
|3 |       3 |      1 |       2 |     0 |     3  |
=============================================

APPLY RESULT:
=============================================
|u |     var |    low |    high |  flag |  level |
|0 |       5 |        |         |     0 |     5  |
|1 |       5 |        |         |     0 |     5  |
|2 |       4 |      0 |       1 |     0 |     4  |
|3 |       3 |      0 |       2 |     0 |     3  |
|4 |       4 |      1 |       0 |     0 |     4  |
|5 |       3 |      1 |       4 |     0 |     3  |
|6 |       2 |      5 |       3 |     0 |     2  |
|7 |       1 |      6 |       3 |     0 |     1  |
=============================================
Time for Apply = 52.000000 ms
Done.
```

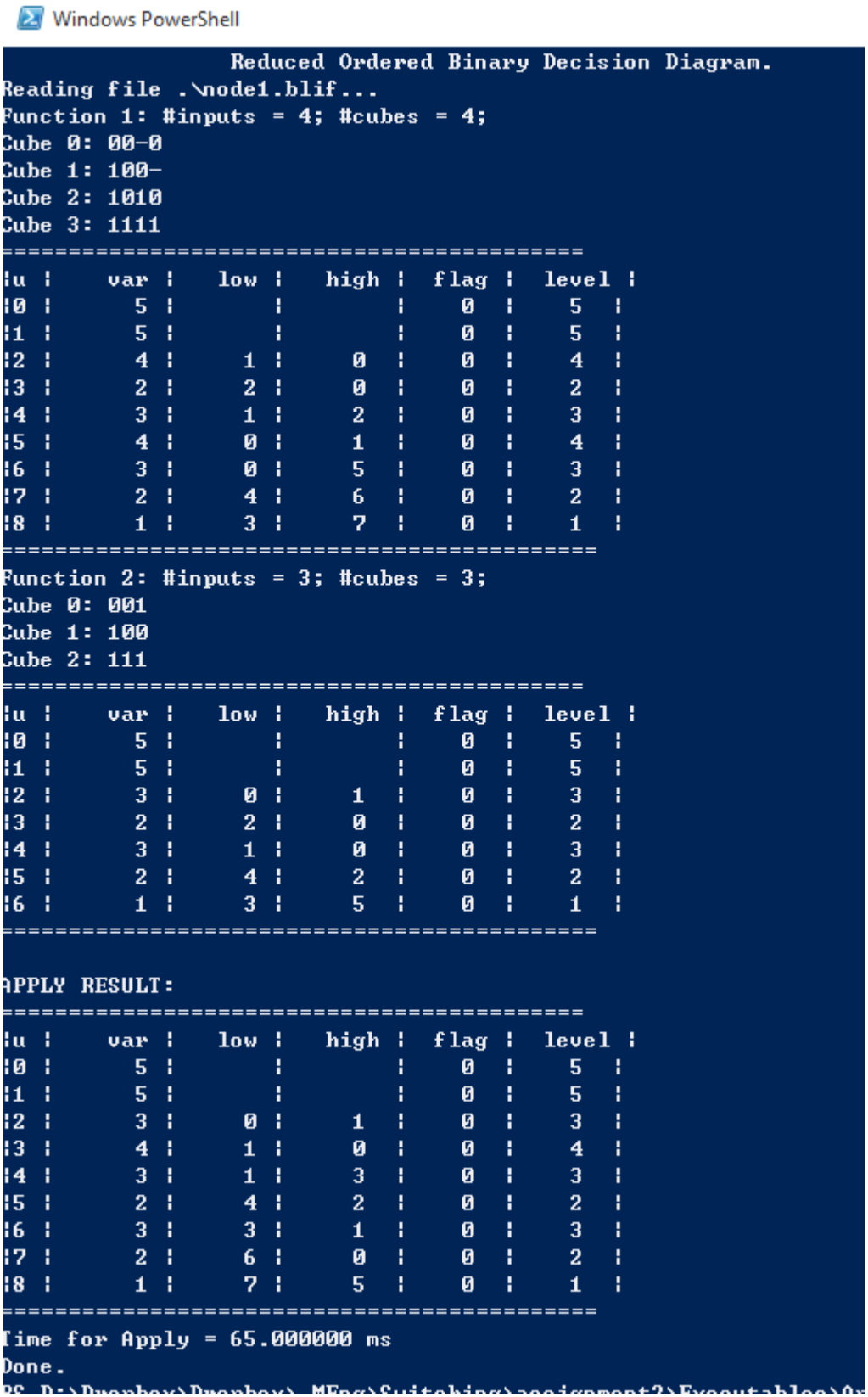
2. **node in original node2.blif <op> node in original node1.blif (With no don't-care sets). The runtimes are 92ms, 65ms and 72ms.**

```
.model top
.inputs x1 x2 x3 x4
.outputs f

.names x1 x2 x3 x4 f
00-0 1
100- 1
1010 1
1111 1

.names x1 x2 x3 f
001 1
100 1
111 1

.end
```

- **node2 AND node1**

```
Reading file .\node1.blif...
Function 1: #inputs = 4; #cubes = 4;
Cube 0: 00-0
Cube 1: 100-
Cube 2: 1010
Cube 3: 1111
================================================
|u |     var |    low |    high |   flag |   level |
|0 |       5 |        |         |      0 |      5  |
|1 |       5 |        |         |      0 |      5  |
|2 |       4 |      1 |       0 |      0 |      4  |
|3 |       2 |      2 |       0 |      0 |      2  |
|4 |       3 |      1 |       2 |      0 |      3  |
|5 |       4 |      0 |       1 |      0 |      4  |
|6 |       3 |      0 |       5 |      0 |      3  |
|7 |       2 |      4 |       6 |      0 |      2  |
|8 |       1 |      3 |       7 |      0 |      1  |
================================================
Function 2: #inputs = 3; #cubes = 3;
Cube 0: 001
Cube 1: 100
Cube 2: 111
================================================
|u |     var |    low |    high |   flag |   level |
|0 |       5 |        |         |      0 |      5  |
|1 |       5 |        |         |      0 |      5  |
|2 |       3 |      0 |       1 |      0 |      3  |
|3 |       2 |      2 |       0 |      0 |      2  |
|4 |       3 |      1 |       0 |      0 |      3  |
|5 |       2 |      4 |       2 |      0 |      2  |
|6 |       1 |      3 |       5 |      0 |      1  |
================================================

APPLY RESULT:
================================================
|u |     var |    low |    high |   flag |   level |
|0 |       5 |        |         |      0 |      5  |
|1 |       5 |        |         |      0 |      5  |
|2 |       4 |      0 |       1 |      0 |      4  |
|3 |       3 |      0 |       2 |      0 |      3  |
|4 |       3 |      1 |       0 |      0 |      3  |
|5 |       2 |      4 |       3 |      0 |      2  |
|6 |       4 |      1 |       0 |      0 |      4  |
|7 |       3 |      0 |       6 |      0 |      3  |
|8 |       2 |      7 |       0 |      0 |      2  |
|9 |       1 |      8 |       5 |      0 |      1  |
================================================
Time for Apply = 92.000000 ms
Done.
```

- **node2 OR node1**



Nu	var	low	high	flag	level
0	5			0	5
1	5			0	5
2	4	1	0	0	4
3	2	2	0	0	2
4	3	1	2	0	3
5	4	0	1	0	4
6	3	0	5	0	3
7	2	4	6	0	2
8	1	3	7	0	1

Nu	var	low	high	flag	level
0	5			0	5
1	5			0	5
2	3	0	1	0	3
3	2	2	0	0	2
4	3	1	0	0	3
5	2	4	2	0	2
6	1	3	5	0	1

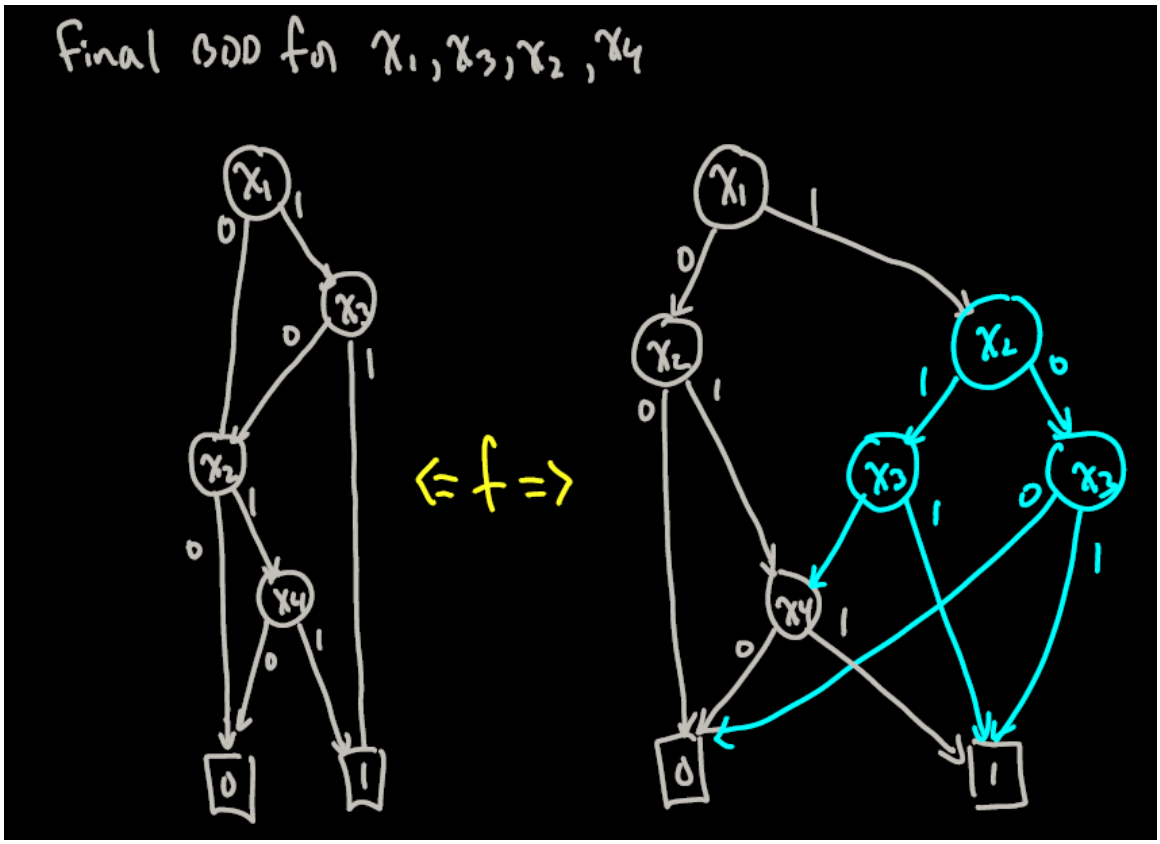
Nu	var	low	high	flag	level
0	5			0	5
1	5			0	5
2	3	0	1	0	3
3	4	1	0	0	4
4	3	1	3	0	3
5	2	4	2	0	2
6	3	3	1	0	3
7	2	6	0	0	2
8	1	7	5	0	1

- **node2 XOR node1**

```
PS D:\Dropbox\Dropbox\_MEng\Switching\assignment2\Executables\Apply> .
                ECE1733 - Switching Theory - Assignment 2
                 Reduced Ordered Binary Decision Diagram.
Reading file .\node1.blif...
Function 1: #inputs = 4; #cubes = 4;
Cube 0: 00-0
Cube 1: 100-
Cube 2: 1010
Cube 3: 1111
=============================================
|u |    var |   low |   high |  flag |  level |
|0 |      5 |       |        |    0  |    5   |
|1 |      5 |       |        |    0  |    5   |
|2 |      4 |     1 |      0 |    0  |    4   |
|3 |      2 |     2 |      0 |    0  |    2   |
|4 |      3 |     1 |      2 |    0  |    3   |
|5 |      4 |     0 |      1 |    0  |    4   |
|6 |      3 |     0 |      5 |    0  |    3   |
|7 |      2 |     4 |      6 |    0  |    2   |
|8 |      1 |     3 |      7 |    0  |    1   |
=============================================
Function 2: #inputs = 3; #cubes = 3;
Cube 0: 001
Cube 1: 100
Cube 2: 111
=============================================
|u |    var |   low |   high |  flag |  level |
|0 |      5 |       |        |    0  |    5   |
|1 |      5 |       |        |    0  |    5   |
|2 |      3 |     0 |      1 |    0  |    3   |
|3 |      2 |     2 |      0 |    0  |    2   |
|4 |      3 |     1 |      0 |    0  |    3   |
|5 |      2 |     4 |      2 |    0  |    2   |
|6 |      1 |     3 |      5 |    0  |    1   |
=============================================

APPLY RESULT:
=============================================
|u |    var |   low |   high |  flag |  level |
|0 |      5 |       |        |    0  |    5   |
|1 |      5 |       |        |    0  |    5   |
|2 |      4 |     1 |      0 |    0  |    4   |
|3 |      3 |     0 |      2 |    0  |    3   |
|4 |      4 |     0 |      1 |    0  |    4   |
|5 |      3 |     2 |      4 |    0  |    3   |
|6 |      2 |     5 |      0 |    0  |    2   |
|7 |      1 |     6 |      3 |    0  |    1   |
=============================================
Time for Apply = 72.000000 ms
Done.
```

Test Results sifting.exe:

1. **node in node6 is the example in the lecture slides (Page 43 to Page 44 in lecture1-7.pdf). The BDD on the left hand side should be the result of the sifting algorithm.**



And the testing result for this node is:

```
PS D:\Dropbox\Dropbox\_MEng\Switching\assignment2\Executables\Sifting
                ECE1733 - Switching Theory - Assignment 2
                 Reduced Ordered Binary Decision Diagram.
Reading file .\node6.blif...
Function 1: #inputs = 4; #cubes = 2;
Cube 0: -1-1
Cube 1: 1-1-

Before sifting, BDD size is 8
============================================
|u |    var |   low |   high |  flag |  level |
|0 |      5 |       |        |    0  |    5   |
|1 |      5 |       |        |    0  |    5   |
|2 |      4 |     0 |      1 |    0  |    3   |
|3 |      2 |     0 |      2 |    0  |    1   |
|4 |      3 |     0 |      1 |    0  |    2   |
|5 |      3 |     2 |      1 |    0  |    2   |
|6 |      2 |     4 |      5 |    0  |    1   |
|7 |      1 |     3 |      6 |    0  |    0   |
============================================
The variable order is: X1 X2 X3 X4

SUMMARY:
One of the Best Options: X2 at No.3 position, and BDD size is 6
One of the Best Orders: X1 X3 X2 X4
After Sifting
============================================
|u |    var |   low |   high |  flag |  level |
|0 |      5 |       |        |    0  |    5   |
|1 |      5 |       |        |    0  |    5   |
|2 |      4 |     0 |      1 |    0  |    3   |
|3 |      2 |     0 |      2 |    0  |    2   |
|4 |      3 |     3 |      1 |    0  |    1   |
|5 |      1 |     3 |      4 |    0  |    0   |
============================================
The time for sifting is 59.000000 ms
Done.
```

As you can see, the result match the result in the lecture slides and the runtime is

59ms.

2. node1.blif can not be optimized as shown below. And the runtime is 59ms.

```
PS D:\Dropbox\Dropbox\_MEng\Switching\assignment2\Executables\
                ECE1733 - Switching Theory - Assignment 2
                 Reduced Ordered Binary Decision Diagram.
Reading file .\node1.blif...
Function 1: #inputs = 3; #cubes = 3;
Cube 0: 001
Cube 1: 100
Cube 2: 111

Before sifting, BDD size is 7
=============================================
|u |    var |   low |   high |  flag |  level |
|0 |      4 |       |        |    0  |    4   |
|1 |      4 |       |        |    0  |    4   |
|2 |      3 |     0 |      1 |    0  |    2   |
|3 |      2 |     2 |      0 |    0  |    1   |
|4 |      3 |     1 |      0 |    0  |    2   |
|5 |      2 |     4 |      2 |    0  |    1   |
|6 |      1 |     3 |      5 |    0  |    0   |
=============================================
The variable order is: X1 X2 X3

SUMMARY:
One of the Best Options: X1 at No.1 position, and BDD size is
One of the Best Orders: X1 X2 X3
After Sifting
=============================================
|u |    var |   low |   high |  flag |  level |
|0 |      4 |       |        |    0  |    4   |
|1 |      4 |       |        |    0  |    4   |
|2 |      3 |     0 |      1 |    0  |    2   |
|3 |      2 |     2 |      0 |    0  |    1   |
|4 |      3 |     1 |      0 |    0  |    2   |
|5 |      2 |     4 |      2 |    0  |    1   |
|6 |      1 |     3 |      5 |    0  |    0   |
=============================================
The time for sifting is 59.000000 ms
Done.
```

3. node2.blif can be optimized as following, the BDD size is reduced by 1. The runtime is 70ms.

```
PS D:\Dropbox\Dropbox\_MEng\Switching\assignment2\Executables\Sifting> .
                ECE1733 - Switching Theory - Assignment 2
                 Reduced Ordered Binary Decision Diagram.
Reading file .\node2.blif...
Function 1: #inputs = 4; #cubes = 4;
Cube 0: 00-0
Cube 1: 100-
Cube 2: 1010
Cube 3: 1111

Before sifting, BDD size is 9
=========================================
|u |    var |   low |   high |  flag |  level |
|0 |      5 |       |        |    0  |    5   |
|1 |      5 |       |        |    0  |    5   |
|2 |      4 |     1 |      0 |    0  |    3   |
|3 |      2 |     2 |      0 |    0  |    1   |
|4 |      3 |     1 |      2 |    0  |    2   |
|5 |      4 |     0 |      1 |    0  |    3   |
|6 |      3 |     0 |      5 |    0  |    2   |
|7 |      2 |     4 |      6 |    0  |    1   |
|8 |      1 |     3 |      7 |    0  |    0   |
=========================================
The variable order is: X1 X2 X3 X4

SUMMARY:
One of the Best Options: X2 at No.4 position, and BDD size is 8
One of the Best Orders: X1 X3 X4 X2
After Sifting
=========================================
|u |    var |   low |   high |  flag |  level |
|0 |      5 |       |        |    0  |    5   |
|1 |      5 |       |        |    0  |    5   |
|2 |      2 |     1 |      0 |    0  |    3   |
|3 |      2 |     0 |      1 |    0  |    3   |
|4 |      4 |     2 |      0 |    0  |    2   |
|5 |      4 |     2 |      3 |    0  |    2   |
|6 |      3 |     2 |      5 |    0  |    1   |
|7 |      1 |     4 |      6 |    0  |    0   |
=========================================
The time for sifting is 70.000000 ms
Done.
```

4. node4.blif can be optimized as following, the BDD size is reduced by 1. The runtime is 67ms.

```
PS D:\Dropbox\Dropbox\_MEng\Switching\assignment2\Executables\Sifting> .\sifti
                ECE1733 - Switching Theory - Assignment 2
                 Reduced Ordered Binary Decision Diagram.
Reading file .\node4.blif...
Function 1: #inputs = 4; #cubes = 4;
Cube 0: 00-0
Cube 1: 100-
Cube 2: -010
Cube 3: 1111

Before sifting, BDD size is 9
=============================================
|u |    var |   low |   high |  flag |  level |
|0 |      5 |       |        |    0  |    5   |
|1 |      5 |       |        |    0  |    5   |
|2 |      4 |     1 |      0 |    0  |    3   |
|3 |      2 |     2 |      0 |    0  |    1   |
|4 |      3 |     1 |      2 |    0  |    2   |
|5 |      4 |     0 |      1 |    0  |    3   |
|6 |      3 |     0 |      5 |    0  |    2   |
|7 |      2 |     4 |      6 |    0  |    1   |
|8 |      1 |     3 |      7 |    0  |    0   |
=============================================
The variable order is: X1 X2 X3 X4

SUMMARY:
One of the Best Options: X2 at No.4 position, and BDD size is 8
One of the Best Orders: X1 X3 X4 X2
After Sifting
=============================================
|u |    var |   low |   high |  flag |  level |
|0 |      5 |       |        |    0  |    5   |
|1 |      5 |       |        |    0  |    5   |
|2 |      2 |     1 |      0 |    0  |    3   |
|3 |      2 |     0 |      1 |    0  |    3   |
|4 |      4 |     2 |      0 |    0  |    2   |
|5 |      4 |     2 |      3 |    0  |    2   |
|6 |      3 |     2 |      5 |    0  |    1   |
|7 |      1 |     4 |      6 |    0  |    0   |
=============================================
The time for sifting is 67.000000 ms
Done.
```

5. node5.blif can not be optimized and the runtime is 70ms

```
PS D:\Dropbox\Dropbox\_MEng\Switching\assignment2\Executables\Sifting> .\s
                ECE1733 - Switching Theory - Assignment 2
                 Reduced Ordered Binary Decision Diagram.
Reading file .\node5.blif...
Function 1: #inputs = 4; #cubes = 4;
Cube 0: --00
Cube 1: 110-
Cube 2: 1-11
Cube 3: 10-0

Before sifting, BDD size is 9
=============================================
|u |    var |   low |   high |  flag |  level |
|0 |      5 |       |        |    0  |    5   |
|1 |      5 |       |        |    0  |    5   |
|2 |      4 |     1 |      0 |    0  |    3   |
|3 |      3 |     2 |      0 |    0  |    2   |
|4 |      3 |     2 |      1 |    0  |    2   |
|5 |      4 |     0 |      1 |    0  |    3   |
|6 |      3 |     1 |      5 |    0  |    2   |
|7 |      2 |     4 |      6 |    0  |    1   |
|8 |      1 |     3 |      7 |    0  |    0   |
=============================================
The variable order is: X1 X2 X3 X4

SUMMARY:
One of the Best Options: X1 at No.1 position, and BDD size is 9
One of the Best Orders: X1 X2 X3 X4
After Sifting
=============================================
|u |    var |   low |   high |  flag |  level |
|0 |      5 |       |        |    0  |    5   |
|1 |      5 |       |        |    0  |    5   |
|2 |      4 |     1 |      0 |    0  |    3   |
|3 |      3 |     2 |      0 |    0  |    2   |
|4 |      3 |     2 |      1 |    0  |    2   |
|5 |      4 |     0 |      1 |    0  |    3   |
|6 |      3 |     1 |      5 |    0  |    2   |
|7 |      2 |     4 |      6 |    0  |    1   |
|8 |      1 |     3 |      7 |    0  |    0   |
=============================================
The time for sifting is 70.000000 ms
Done.
```

Limitations:

- For apply.exe, it only supports AND, OR and XOR operations for now.
- The program will not draw the BDD based on the T table, therefore the result is not obvious to the user.
- And the sifting algorithm can not process the node3.blif provided by Professor Brown due to some memory issue.
- Currently both sifting and apply implementation are not taking don't-care set as input.

Conclusion:

We successfully implemented the software program for sifting and apply algorithm, and during the development process, we have gained a lot of experience on manipulating the BDDs by hand.