

Assignment 01

Truth table given in the Student ID Generated PDF

X1	X2	X3	X4	Z1	Z2
0	0	0	0	1	0
0	0	0	1	1	0
0	0	1	0	0	1
0	0	1	1	1	1
0	1	0	0	1	0
0	1	0	1	1	1
0	1	1	0	0	0
0	1	1	1	1	1
1	0	0	0	0	1
1	0	0	1	1	1
1	0	1	0	1	0
1	0	1	1	0	1
1	1	0	0	1	1
1	1	0	1	0	1
1	1	1	0	1	0
1	1	1	1	0	1

Answer 1.1:

From the truth table, I analysed the matchups for X1, X2, X3 and X4 which result in "1" for Z1 and Z2 outputs respectively. (For example, when X1 = 0, X2 = 0, X3 = 0, X4 = 0, Z1 = 1, therefore $\overline{X1} \overline{X2} \overline{X3} \overline{X4}$ is included)
Then I recorded the matchups in Sum of Products (SOP) notation.

$$Z1(X1, X2, X3, X4) = \overline{X1} \overline{X2} \overline{X3} \overline{X4} + \overline{X1} \overline{X2} \overline{X3} X4 + \overline{X1} \overline{X2} X3 X4 + \overline{X1} X2 \overline{X3} \overline{X4} + \overline{X1} X2 \overline{X3} X4 + \overline{X1} X2 X3 X4 + X1 \overline{X2} \overline{X3} X4 + X1 \overline{X2} X3 \overline{X4} + X1 X2 \overline{X3} \overline{X4} + X1 X2 X3 \overline{X4}$$

$$Z2(X1, X2, X3, X4) = \overline{X1} \overline{X2} X3 \overline{X4} + \overline{X1} \overline{X2} X3 X4 + \overline{X1} X2 \overline{X3} X4 + \overline{X1} X2 X3 X4 + X1 \overline{X2} \overline{X3} \overline{X4} + X1 \overline{X2} \overline{X3} X4 + X1 \overline{X2} X3 \overline{X4} + X1 \overline{X2} X3 X4 + X1 X2 \overline{X3} \overline{X4} + X1 X2 \overline{X3} X4 + X1 X2 X3 \overline{X4}$$

Answer 1.2:

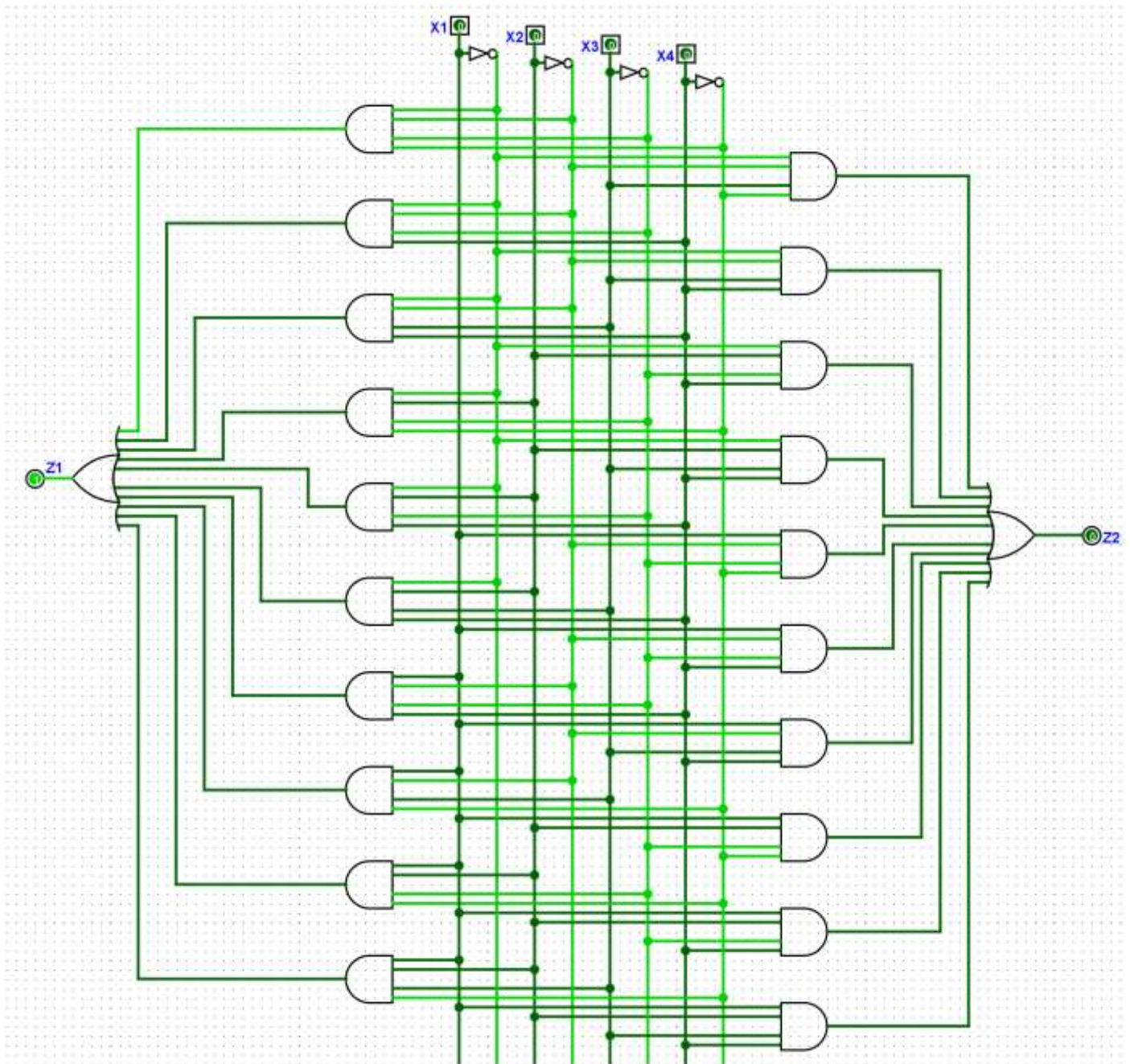
From the Boolean expression, I constructed AND gates using the products respectively, as only those specific inputs of X1, X2, X3 and X4 result in a "1" for Z1 and Z2. After laying out all the "1" input assemblies, I use OR gates to show the results of Z1 and Z2 respectively. The inputs for $\overline{X1}$, $\overline{X2}$, $\overline{X3}$ and $\overline{X4}$ were constructed using NOT gates.

Name: Diong Chen Xi
Student ID: 32722656

Class Number: FIT1047
Tutor Name: Dr. Tan Chee Keong

Circuit of Z1 and Z2

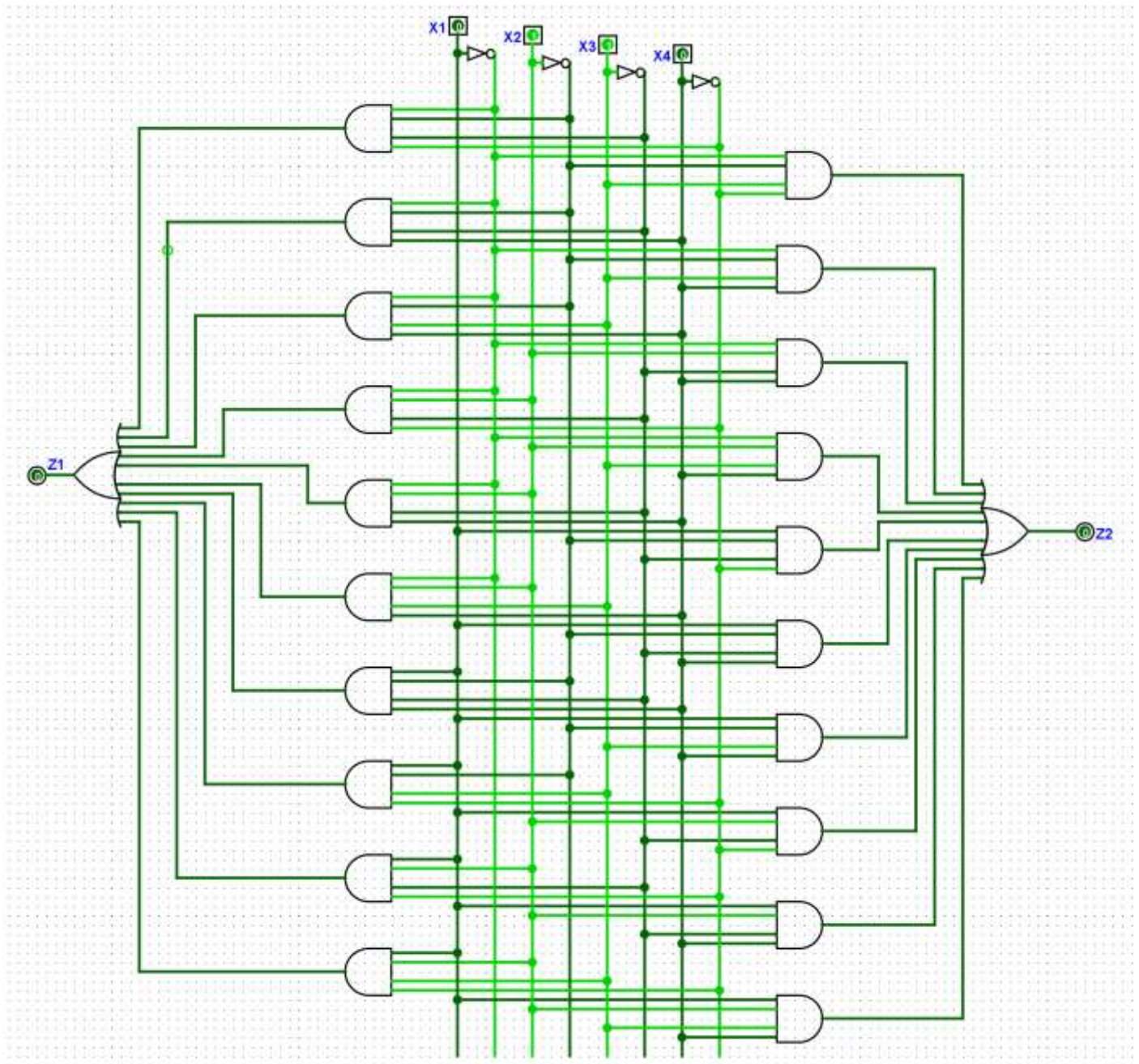
Input: $X1 = 0, X2 = 0, X3 = 0, X4 = 0$ Output: $Z1 = 1, Z2 = 0$



Name: Diong Chen Xi
Student ID: 32722656

Class Number: FIT1047
Tutor Name: Dr. Tan Chee Keong

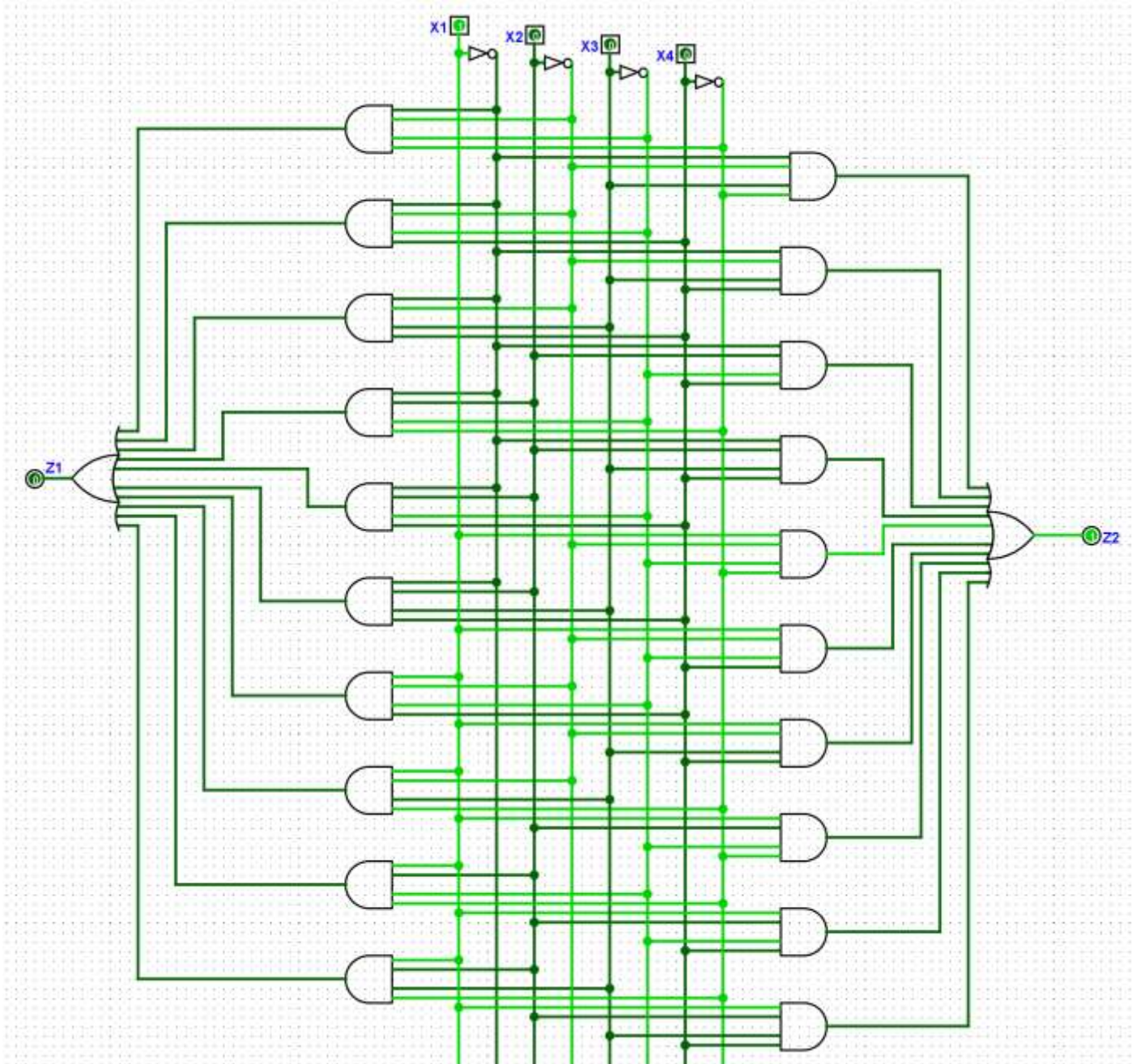
Input: $X_1 = 0, X_2 = 1, X_3 = 1, X_4 = 0$ Output: $Z_1 = 0, Z_2 = 0$



Name: Diong Chen Xi
Student ID: 32722656

Class Number: FIT1047
Tutor Name: Dr. Tan Chee Keong

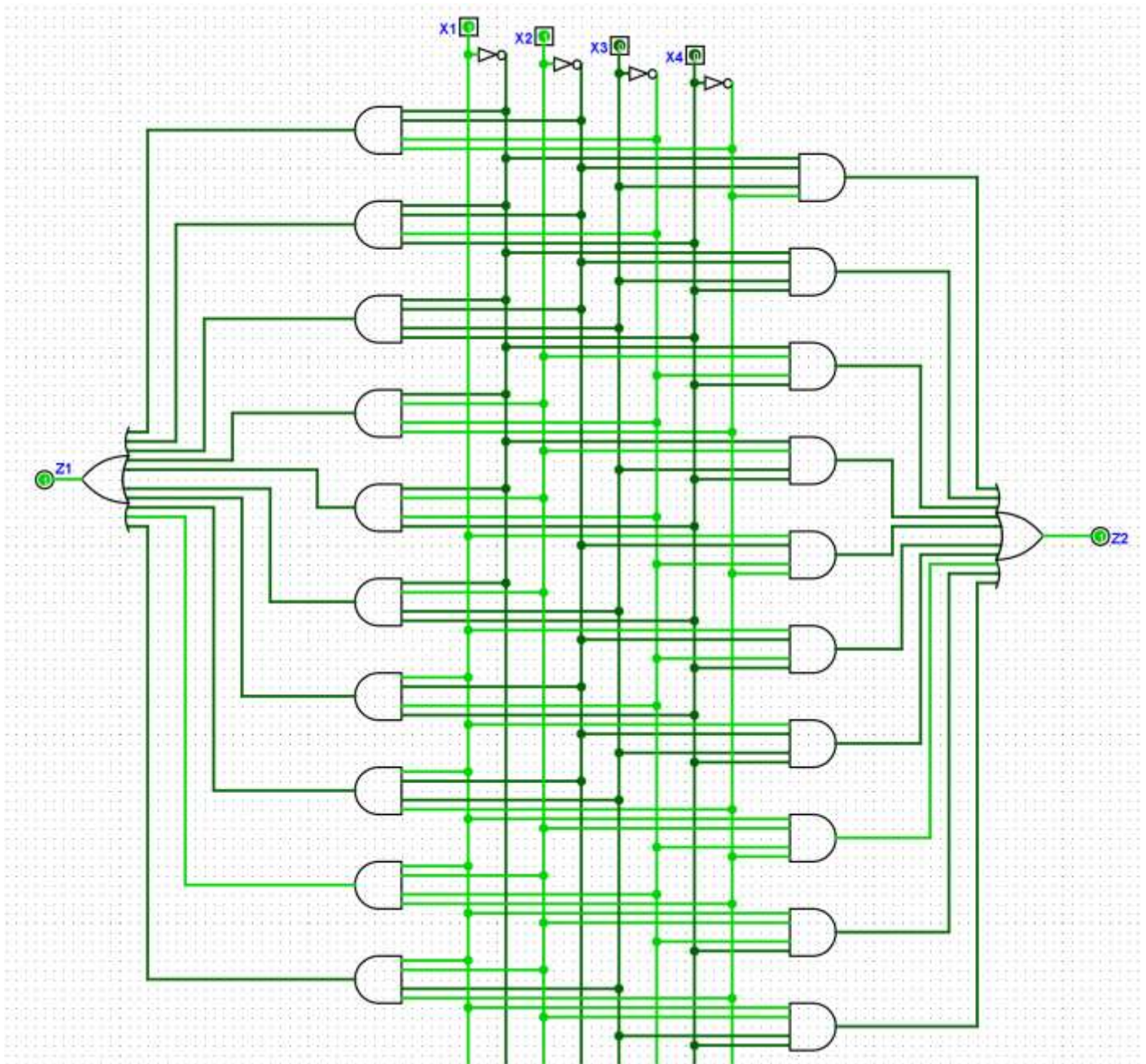
Input: $X_1 = 1, X_2 = 0, X_3 = 0, X_4 = 0$ Output: $Z_1 = 0, Z_2 = 1$



Name: Diong Chen Xi
Student ID: 32722656

Class Number: FIT1047
Tutor Name: Dr. Tan Chee Keong

Input: $X_1 = 1, X_2 = 1, X_3 = 0, X_4 = 0$ Output: $Z_1 = 1, Z_2 = 1$



Answer 1.3:

Z1 Subtask1 Karnaugh Map simplification

$x_3 \backslash x_4$ $x_1 \backslash x_2$	00	01	11	10
00	1	1	1	0
01	1	1	1	0
11	1	0	0	1
10	0	1	0	1

$$Z = \bar{x}_1 \bar{x}_3 + x_2 \bar{x}_3 \bar{x}_4 + \bar{x}_2 \bar{x}_3 x_4 + \bar{x}_1 x_4 + x_1 x_3 \bar{x}_4$$

Z1 Subtask2 Boolean Identities simplification

$$Z = \bar{x}_1 \bar{x}_2 \bar{x}_3 \bar{x}_4 + \bar{x}_1 \bar{x}_2 \bar{x}_3 x_4 + \bar{x}_1 \bar{x}_2 x_3 x_4 + \bar{x}_1 x_2 \bar{x}_3 \bar{x}_4 + \bar{x}_1 x_2 \bar{x}_3 x_4 + \bar{x}_1 x_2 x_3 x_4 + x_1 \bar{x}_2 \bar{x}_3 x_4 + x_1 \bar{x}_2 x_3 \bar{x}_4 + x_1 x_2 \bar{x}_3 \bar{x}_4 + x_1 x_2 x_3 \bar{x}_4$$

$$= \bar{x}_1 \bar{x}_2 \bar{x}_3 \bar{x}_4 + (\bar{x}_1 \bar{x}_2 \bar{x}_3 x_4 + \bar{x}_1 \bar{x}_2 \bar{x}_3 x_4) + \bar{x}_1 \bar{x}_2 x_3 x_4 + (\bar{x}_1 x_2 \bar{x}_3 \bar{x}_4 + \bar{x}_1 x_2 \bar{x}_3 \bar{x}_4) + \bar{x}_1 x_2 \bar{x}_3 x_4 + \bar{x}_1 x_2 x_3 x_4 + x_1 \bar{x}_2 \bar{x}_3 x_4 + x_1 \bar{x}_2 x_3 \bar{x}_4 + x_1 x_2 \bar{x}_3 \bar{x}_4 + x_1 x_2 x_3 \bar{x}_4 \quad \text{[Idempotent OR]}$$

$$= (\bar{x}_1 \bar{x}_2 \bar{x}_3 \bar{x}_4 + \bar{x}_1 \bar{x}_2 \bar{x}_3 x_4) + (\bar{x}_1 \bar{x}_2 \bar{x}_3 x_4 + x_1 \bar{x}_2 \bar{x}_3 x_4) + (\bar{x}_1 \bar{x}_2 x_3 x_4 + \bar{x}_1 x_2 x_3 x_4) + (\bar{x}_1 x_2 \bar{x}_3 \bar{x}_4 + \bar{x}_1 x_2 \bar{x}_3 x_4) + (\bar{x}_1 x_2 \bar{x}_3 \bar{x}_4 + x_1 \bar{x}_2 \bar{x}_3 \bar{x}_4) + (x_1 \bar{x}_2 \bar{x}_3 \bar{x}_4 + x_1 \bar{x}_2 x_3 \bar{x}_4) \quad \text{[Associative OR]}$$

$$= \bar{x}_1 \bar{x}_2 \bar{x}_3 + \bar{x}_2 \bar{x}_3 x_4 + \bar{x}_1 x_3 x_4 + \bar{x}_1 x_2 \bar{x}_3 + x_2 \bar{x}_3 \bar{x}_4 + x_1 x_3 \bar{x}_4 \quad \text{[Inverse OR]}$$

$$= (\bar{x}_1 \bar{x}_2 \bar{x}_3 + \bar{x}_1 x_2 \bar{x}_3) + \bar{x}_1 x_3 x_4 + x_1 x_3 \bar{x}_4 + \bar{x}_2 \bar{x}_3 x_4 + x_2 \bar{x}_3 \bar{x}_4 \quad \text{[Associative OR]}$$

$$= \bar{x}_1 \bar{x}_3 + \bar{x}_1 x_3 x_4 + x_1 x_3 \bar{x}_4 + \bar{x}_2 \bar{x}_3 x_4 + x_2 \bar{x}_3 \bar{x}_4 \quad \text{[Inverse OR]}$$

By comparing the results of the K-map method and the Boolean Identities method, I found out that both of the optimised Boolean Function have equal amount of terms. The terms are also similar in grouping.

Z2 Subtask1 Karnaugh Map simplification

x_3x_4 x_1x_2	00	01	11	10
00	0	0	1	1
01	0	1	1	0
11	1	1	1	0
10	1	1	1	0

$$Z_2 = x_1 \bar{x}_3 + x_1 x_4 + x_2 x_4 + \bar{x}_1 \bar{x}_2 x_3$$

Z2 Subtask2 Boolean Identities simplification

$$Z_2 = \bar{x}_1 \bar{x}_2 x_3 \bar{x}_4 + \bar{x}_1 \bar{x}_2 x_3 x_4 + \bar{x}_1 x_2 \bar{x}_3 \bar{x}_4 + \bar{x}_1 x_2 \bar{x}_3 x_4 + x_1 \bar{x}_2 \bar{x}_3 \bar{x}_4 + x_1 \bar{x}_2 \bar{x}_3 x_4 + x_1 \bar{x}_2 x_3 \bar{x}_4 + x_1 \bar{x}_2 x_3 x_4 + x_1 x_2 \bar{x}_3 \bar{x}_4 + x_1 x_2 \bar{x}_3 x_4 + x_1 x_2 x_3 \bar{x}_4 + x_1 x_2 x_3 x_4$$

$$= \bar{x}_1 \bar{x}_2 x_3 \bar{x}_4 + \bar{x}_1 \bar{x}_2 x_3 x_4 + \bar{x}_1 x_2 \bar{x}_3 \bar{x}_4 + \bar{x}_1 x_2 \bar{x}_3 x_4 + x_1 \bar{x}_2 \bar{x}_3 \bar{x}_4 + (x_1 \bar{x}_2 \bar{x}_3 x_4 + x_1 \bar{x}_2 x_3 \bar{x}_4) + x_1 \bar{x}_2 x_3 x_4 + x_1 x_2 \bar{x}_3 \bar{x}_4 + (x_1 x_2 \bar{x}_3 x_4 + x_1 x_2 \bar{x}_3 \bar{x}_4) + x_1 x_2 x_3 x_4$$

[Idempotent OR]

$$= (\bar{x}_1 \bar{x}_2 x_3 \bar{x}_4 + \bar{x}_1 \bar{x}_2 x_3 x_4) + (\bar{x}_1 x_2 \bar{x}_3 \bar{x}_4 + \bar{x}_1 x_2 \bar{x}_3 x_4) + (x_1 \bar{x}_2 \bar{x}_3 \bar{x}_4 + x_1 \bar{x}_2 \bar{x}_3 x_4) + (x_1 \bar{x}_2 x_3 \bar{x}_4 + x_1 \bar{x}_2 x_3 x_4) + (x_1 x_2 \bar{x}_3 \bar{x}_4 + x_1 x_2 \bar{x}_3 x_4) + (x_1 x_2 x_3 \bar{x}_4 + x_1 x_2 x_3 x_4)$$

[Associative OR]

$$= \bar{x}_1 \bar{x}_2 x_3 + \bar{x}_1 x_2 x_4 + x_1 \bar{x}_2 \bar{x}_3 + x_1 \bar{x}_2 x_4 + x_1 x_2 \bar{x}_3 + x_1 x_2 x_4 \quad \text{[Inverse OR]}$$

$$= \bar{x}_1 \bar{x}_2 x_3 + \bar{x}_1 x_2 x_4 + x_1 \bar{x}_2 \bar{x}_3 + x_1 \bar{x}_2 x_4 + x_1 x_2 \bar{x}_3 + (x_1 x_2 x_4 + x_1 x_2 x_4) \quad \text{[Idempotent OR]}$$

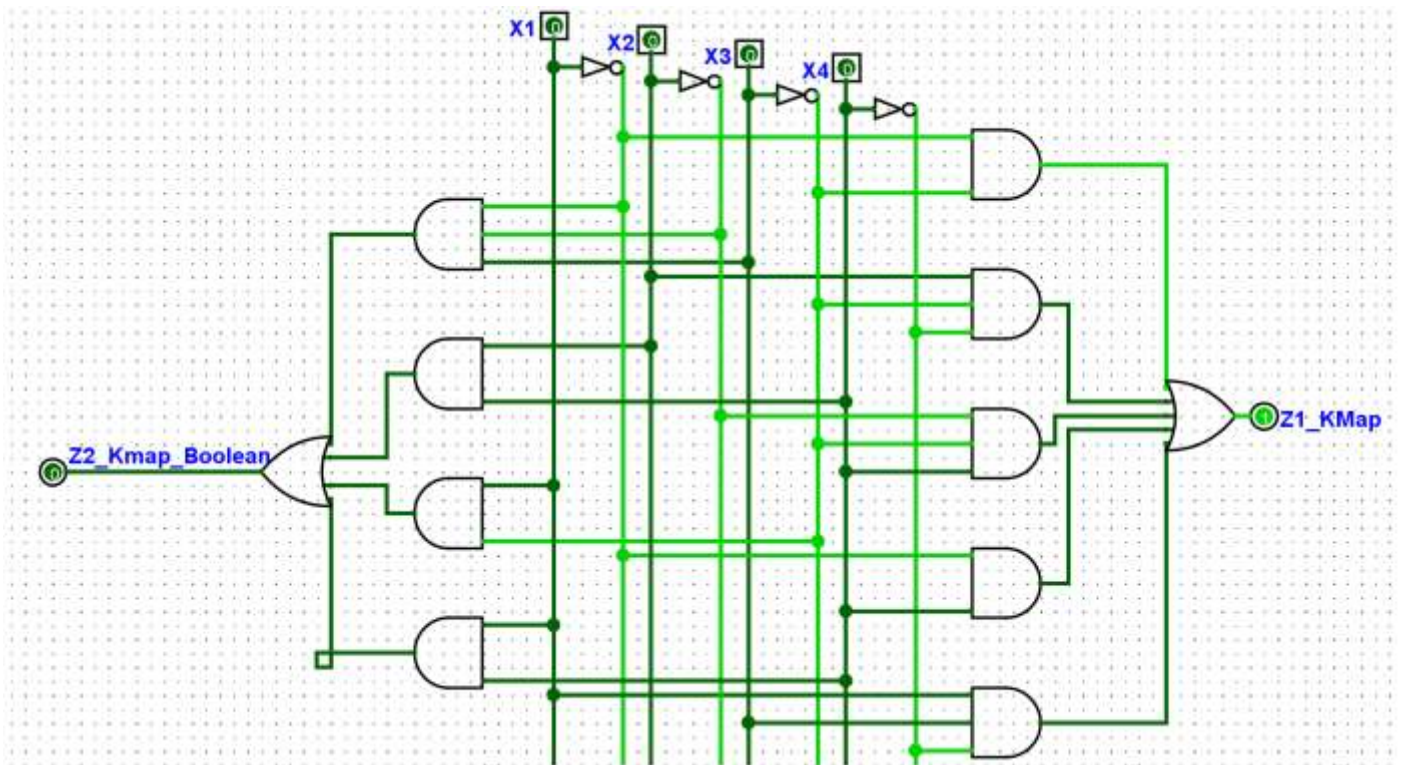
$$= \bar{x}_1 \bar{x}_2 x_3 + (\bar{x}_1 x_2 x_4 + x_1 x_2 x_4) + (x_1 \bar{x}_2 \bar{x}_3 + x_1 x_2 \bar{x}_3) + (x_1 \bar{x}_2 x_4 + x_1 x_2 x_4) \quad \text{[Associative OR]}$$

$$= \bar{x}_1 \bar{x}_2 x_3 + x_2 x_4 + x_1 \bar{x}_3 + x_1 x_4 \quad \text{[Inverse OR]}$$

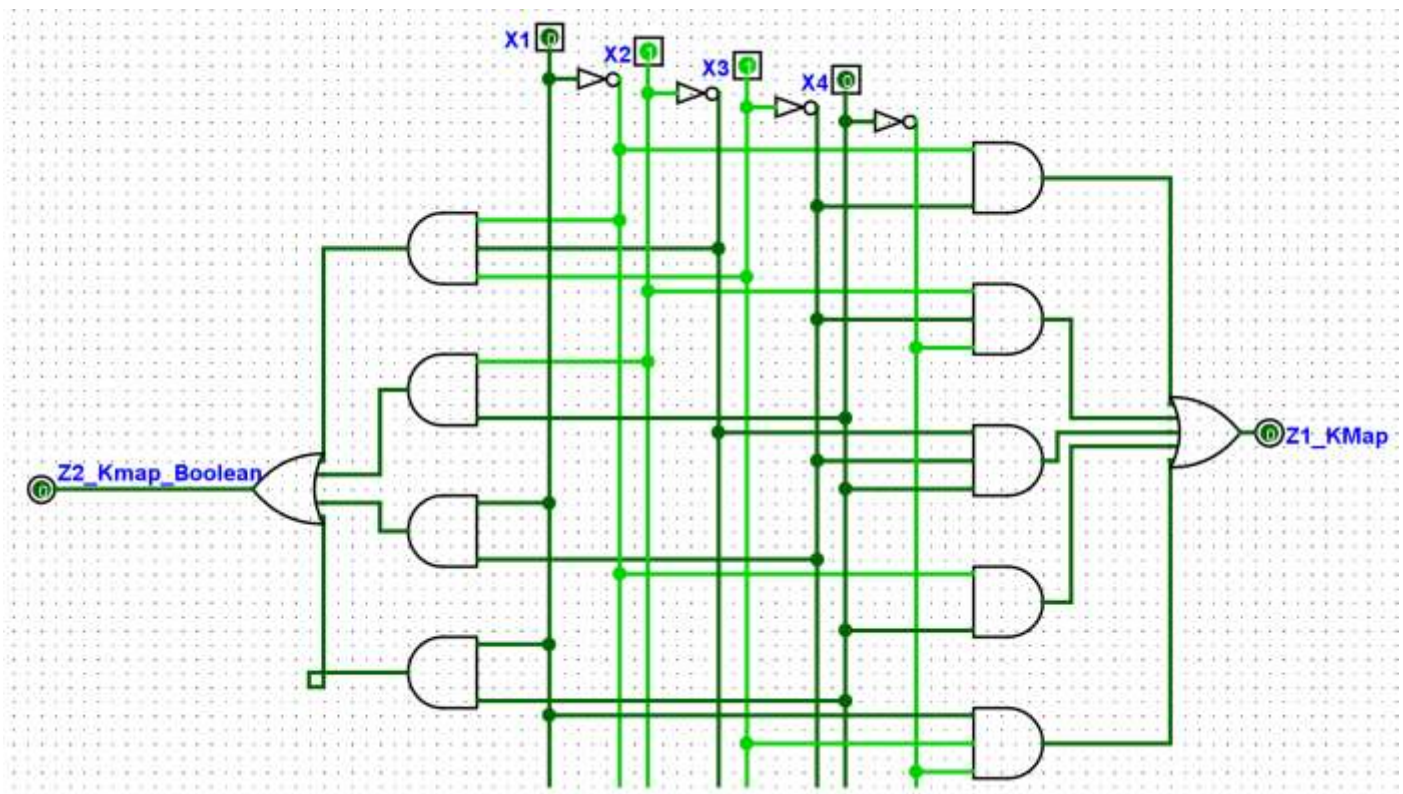
By comparing the results of the K-map method and the Boolean Identities method, I found out that both of the optimised Boolean Function have equal amount of terms. The terms are identical in this case.

Optimized circuits of Z1 using Kmap and Z2

Input: $X1 = 0, X2 = 0, X3 = 0, X4 = 0$ Output: $Z1 = 1, Z2 = 0$



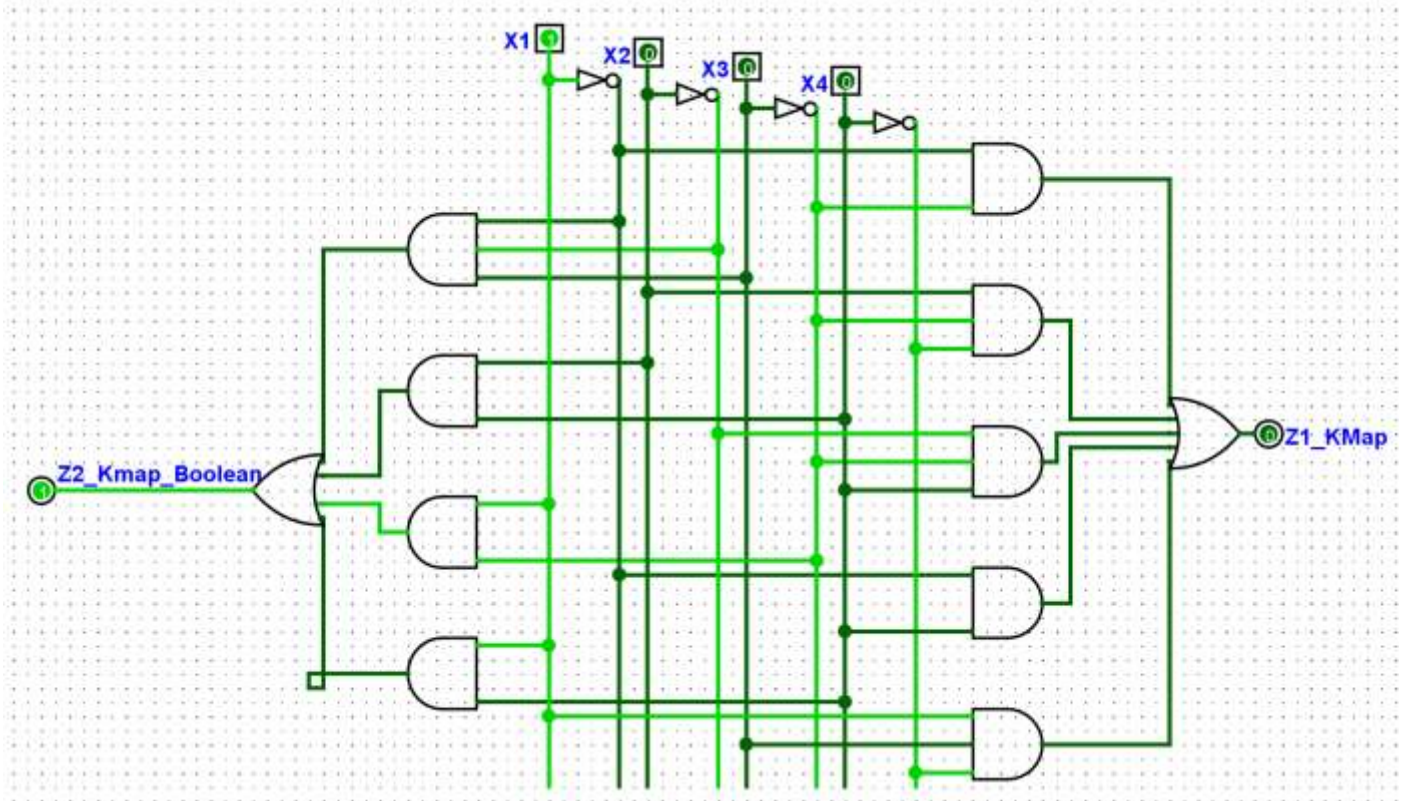
Input: $X1 = 0, X2 = 1, X3 = 1, X4 = 0$ Output: $Z1 = 0, Z2 = 0$



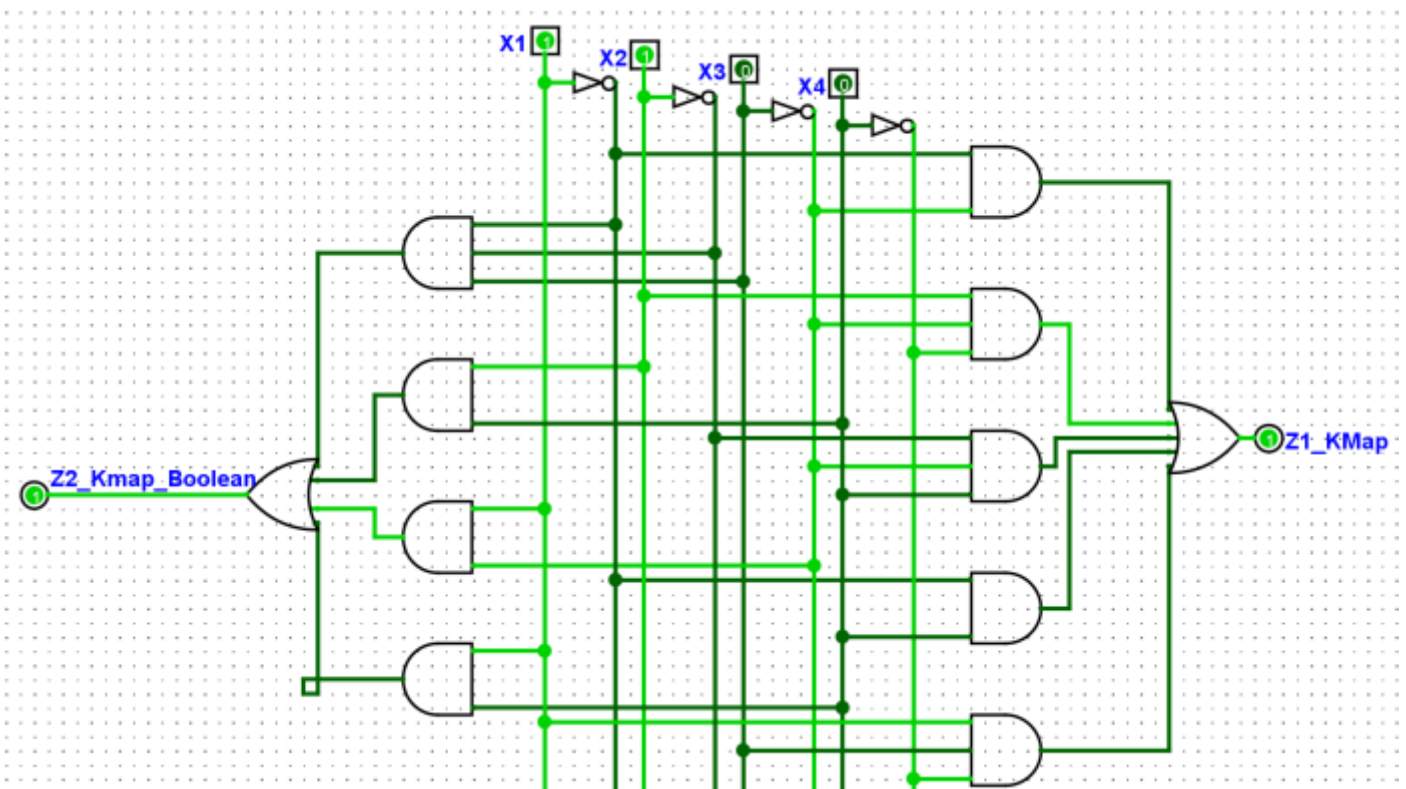
Name: Diong Chen Xi
Student ID: 32722656

Class Number: FIT1047
Tutor Name: Dr. Tan Chee Keong

Input: $X1 = 1, X2 = 0, X3 = 0, X4 = 0$ Output: $Z1 = 0, Z2 = 1$

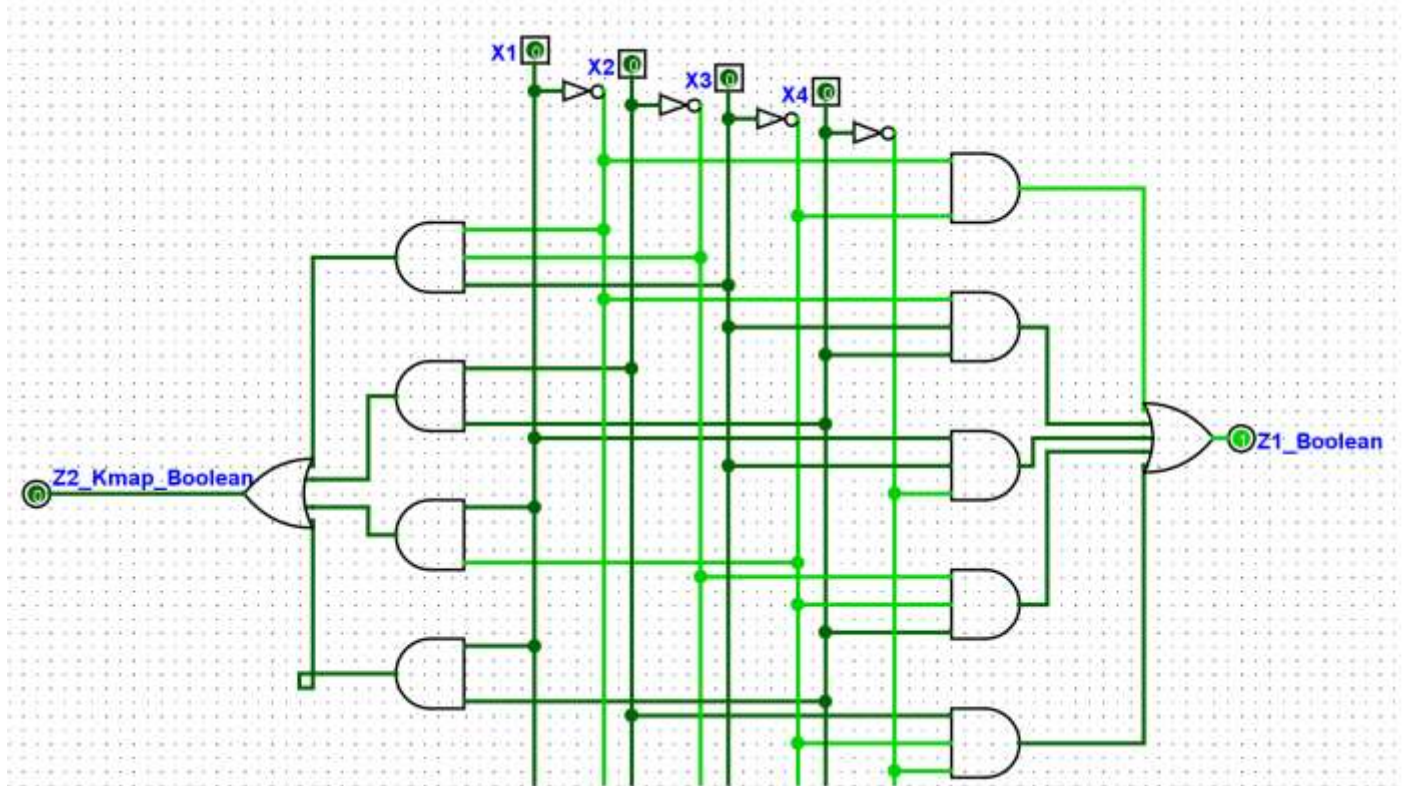


Input: $X1 = 1, X2 = 1, X3 = 0, X4 = 0$ Output: $Z1 = 1, Z2 = 1$

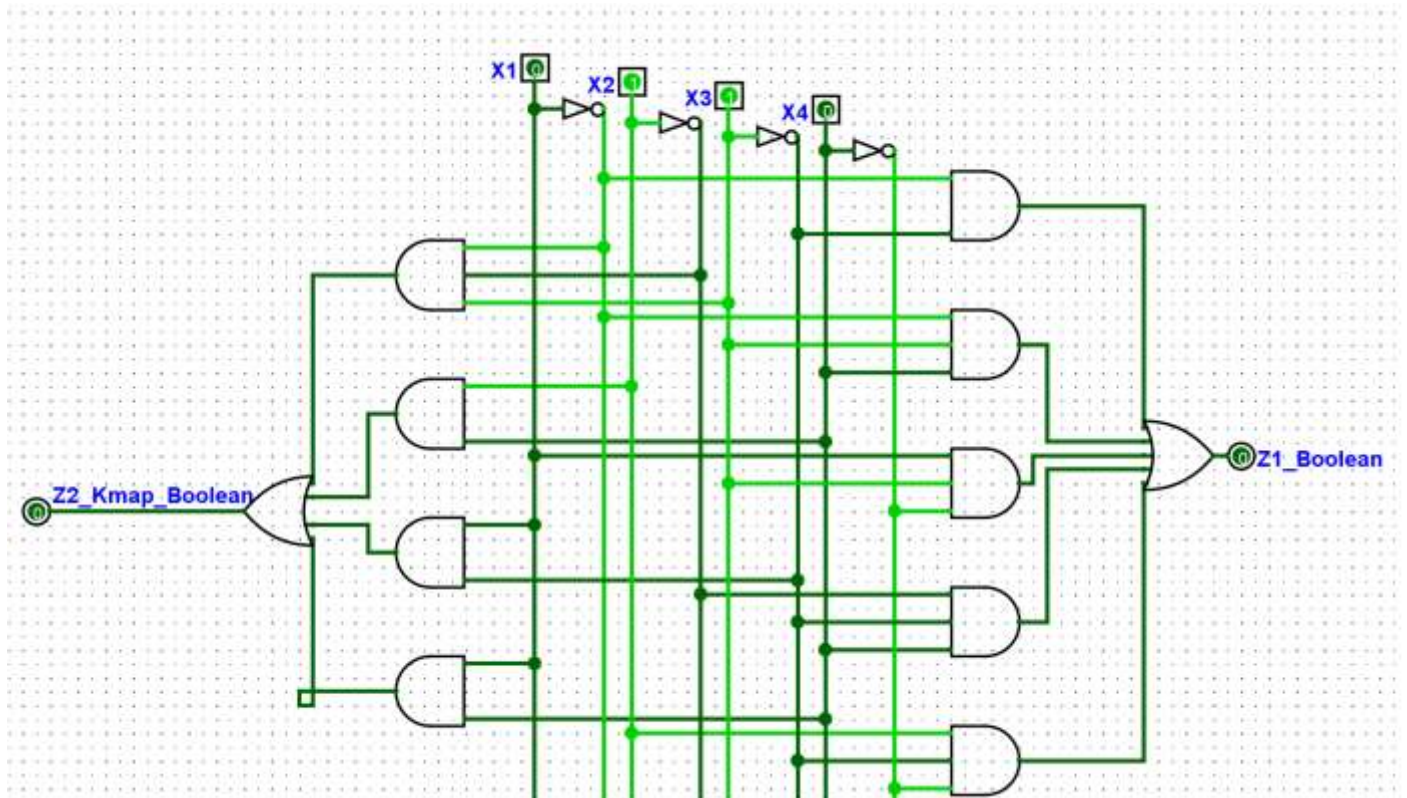


Optimized Circuits of Z1 using Boolean Identities and Z2

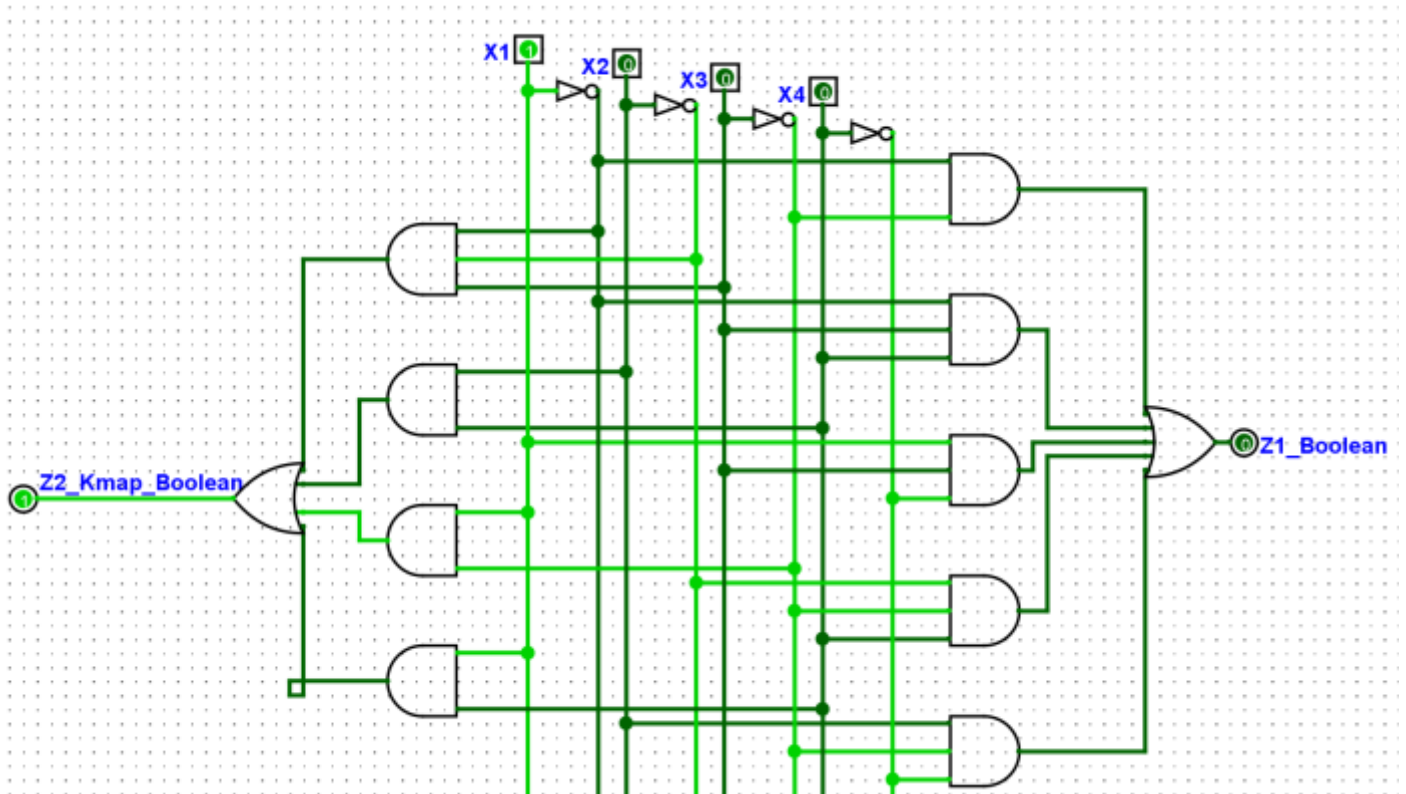
Input: $X1 = 0, X2 = 0, X3 = 0, X4 = 0$ Output: $Z1 = 1, Z2 = 0$



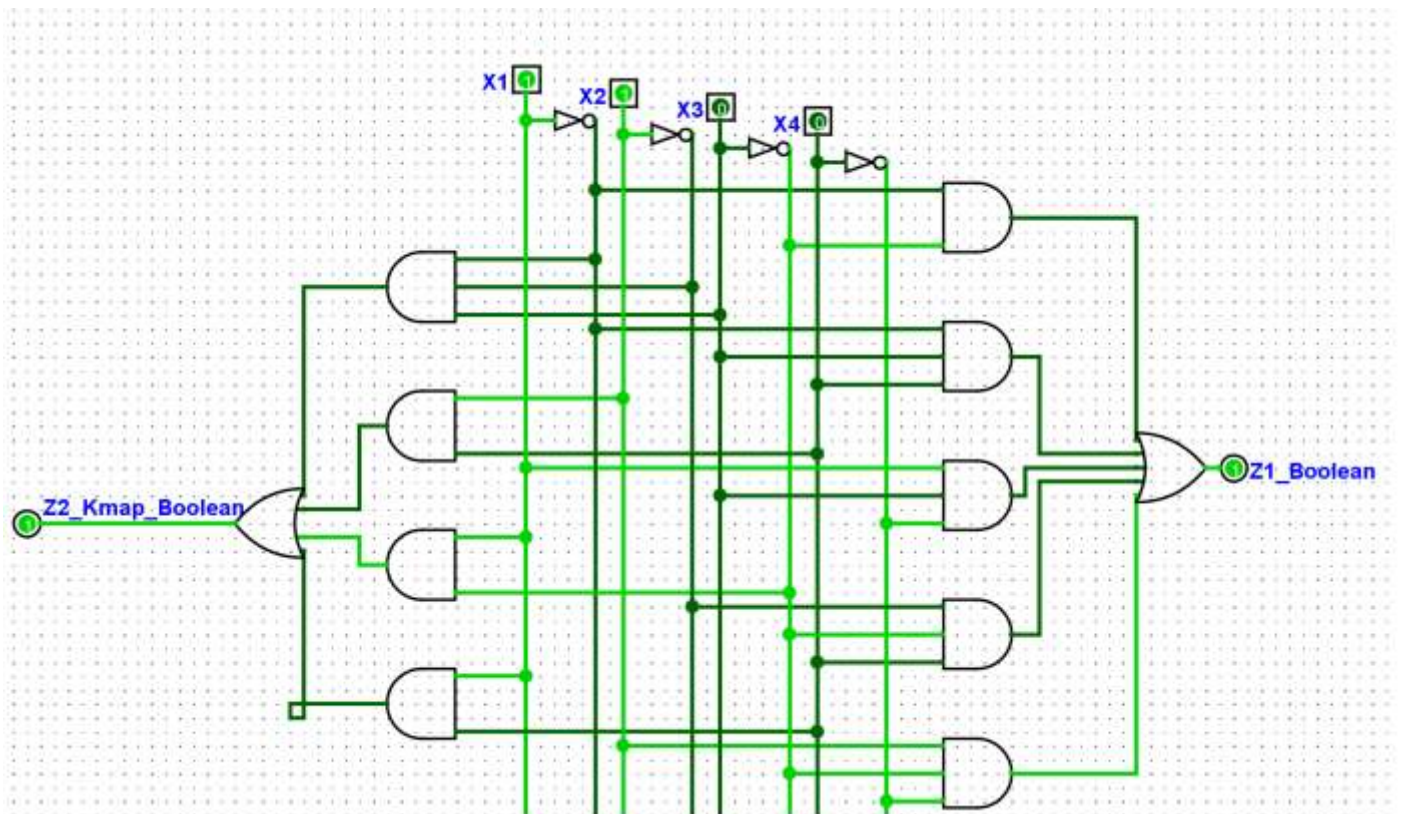
Input: $X1 = 0, X2 = 1, X3 = 1, X4 = 0$ Output: $Z1 = 0, Z2 = 0$



Input: $X1 = 1, X2 = 0, X3 = 0, X4 = 0$ Output: $Z1 = 0, Z2 = 1$



Input: $X1 = 1, X2 = 1, X3 = 0, X4 = 0$ Output: $Z1 = 1, Z2 = 1$



Documentation of Task 2

Task 2.1

Assembly code: Autosaved file

```

1 /Main program
2 firstname, Load count
3 Skipcond 800 /If 5 characters are entered, count will be 0
4 Jump AddComma /Adds a comma if first name is fully inputted
5 JnS subInputNames /otherwise continue receiving input for first name
6 Load count
7 Subt One /after receiving an input, subtract 1 from the counter
8 Store count
9 Jump firstname /continue looping until first name has finished inputting
10 AddComma, Load comma
11 StoreI myNameAdd /Stores comma into current address
12 Load myNameAdd
13 Add One /moves to the next address to store
14 Store myNameAdd
15 Load count
16 Add Five /resets the counter after adding a comma
17 Store count
18 lastname, Load count

```

Machine halted normally.

AC: 0024
IR: 7000
MAR: 051
MBR: 7000
PC: 052
IN: 0024
OUT: 0024

OUTPUT MODE: UNICODE (UTF-16BE) v

Diong ChenX
KangH ongBo
DrTan CheeK
\$

	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
2E0	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
2F0	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
300	0044	0069	006F	006E	0067	002C	0043	0060	0065	006E	0058	002E	0000	0000	0000	0000
310	0040	0061	006E	0067	0040	002C	0067	006E	0067	0042	006E	002E	0000	0000	0000	0000
320	0044	0072	0054	0063	006E	002C	0043	0060	0065	0065	004B	002E	0000	0000	0000	0000
330	0024	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
340	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

From the screenshot, I entered three names as the input, namely Diong ChenX, KangH ongBo and DrTan CheeK (my name, my friend's name, and my tutor's name). I programmed the MARIE code such that the user only needs to input the first name and last name, while the commas and full stops are automatically stored inside the respective addresses.

As we can see, the output is as desired, a space is printed in place of a comma, and a new line is printed in place of a full stop. Also the '\$' sign is printed out as an indication of the end of the program. The name strings are also stored in three different lines of addresses (i.e. 300, 310, and 320) while the '\$' sign is stored in address 330.

Task 2.2

Assembly code:

```

1 /Main program
2 firstname, Load count
3 Skipcond 000 /If 5 characters are entered, count will be 0
4 Jump AddComma /Adds a comma if first name is fully inputted
5 JnS subInputNames /otherwise continue receiving input for first name
6 Load count
7 Subt One /after receiving an input, subtract 1 from the counter
8 Store count
9 Jump firstname /continue looping until first name has finished inputting
10 AddComma, Load comma
11 StoreI myNameAdd /Stores comma into current address
12 Load myNameAdd
13 Add One /moves to the next address to store
14 Store myNameAdd
15 Load count
16 Add Five /resets the counter after adding a comma
17 Store count
18

```

Machine halted normally.

	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
2E0	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
2F0	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
300	0064	0048	004F	004E	0047	0020	00C3	0010	0045	0041	0078	002E	0000	0000	0000	0000
310	0040	0041	004E	0047	0000	0020	0041	004E	0047	0042	0042	002E	0000	0000	0000	0000
320	0064	0052	0078	0041	004E	0020	00C3	0010	0045	0045	0068	002E	0000	0000	0000	0000
330	0024	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
340	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

OUTPUT MODE: UNICODE (UTF-16BE) ▼

Output log: RTL log: Watch list: Input list

AC: 0024

IR: 7000

MAR: 0098

MBR: 7000

PC: 0099

IN: 0024

OUT: 0024

Diong ChenX
KangH ongBo
DrTan CheeK
dIONG cHENx
KANGh ONGbO
dRTAN cHEEK
\$

From the screenshot, I entered three names as the input, namely Diong ChenX, KangH ongBo and DrTan CheeK (my name, my friend's name, and my tutor's name). I programmed the MARIE code such that the user only needs to input the first name and last name, while the commas and full stops are automatically stored inside the respective addresses.

As we can see, the output is as desired, a space is printed in place of a comma, and a new line is printed in place of a full stop. Also the names with switched cases are printed out after the original names, and the '\$' sign is printed out as an indication of the end of the program. The name strings are also stored in three different lines of addresses (i.e. 300, 310, and 320) while the '\$' sign is stored in address 330.

Task 2.3

Assembly code:

```

1 /Saving address of mySubstKey1Addr for later use
2 Load mySubstKey1Addr
3 Store oriadd
4 /Main program
5 inputname, Jn5 subInputNames
6     Jump inputname /continue looping until the name has finished inputting
7 storenextadd, Load myNameAdd
8     Add One /moves to next address for input
9     Store myNameAdd
10    Jump inputname
11 storenextline, Load myNameAdd
12    Add nextrow /moves to next row for input
13    Store myNameAdd
14    Jump inputname
15
16 printfirstname, LoadI PrintNameAdd
17    Subt comma /checks if the current character is a comma
18

```

Machine halted normally.

	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
5F0	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
600	0071	0064	0072	0073	006C	0066	006E	002F	006F	0064	002E	0000	0000	0000	0000	0000
610	0066	006B	006B	0071	0063	002C	0067	006C	0072	0071	006A	002E	0000	0000	0000	0000
620	006E	0064	0071	006A	002C	006B	0072	0073	006A	0065	0072	002E	0000	0000	0000	0000
630	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
640	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
650	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

AC: 0024
IR: 7000
MAR: 099
MBR: 7000
PC: 09A
IN: 0024
OUT: 0024

OUTPUT MODE: UNICODE (UTF-16BE)

patrick lee
chenx diong
kang hongbo
schwulfn ohh
fkha glrq
ndqj krgier
\$

From the screenshot, I entered three names as the input, namely “patrick,lee.” , “chenx,diong.” and “kang,hongbo.”. For this MARIE code I did not program in such a way that the commas and full stops are automatically stored and instead require user input.

As we can see, the output is as desired, a space is printed in place of a comma, and a new line is printed in place of a full stop. Also the names after substitution are printed out after the original names, and the ‘\$’ sign is printed out as an indication of the end of the program. The name strings are also stored in three different lines of addresses (i.e. 300, 310, and 320) while the ‘\$’ sign is stored in address 330. Meanwhile the substituted name strings are stored in other addresses (i.e. 600, 610 and 620).