Department of Computer Science
Faculty of Computing
UNIVERSITI TEKNOLOGI MALAYSIA

SUBJECT NAME: **DIGITAL LOGIC**

SUBJECT CODE: **SECR1013**

SEMESTER: **1**

LAB TITLE: **LAB 2: COMBINATIONAL DIGITAL CIRCUIT DESIGN SIMULATION USING DEEDS SIMULATOR**

GROUP MEMBER : **Name: NAJMA SHAKIRAH BINTI SHAHRULZAMAN**

Email: najmashakirah@graduate.utm.my

Name: CHUA SHANG YEET

Email: chuayeet@graduate.utm.my

SUBMITTED DATE: **21/12/2023**

COMMENTS:

MARKS:

Lab # 2

Combinational Digital Circuit Design Simulation Using Deeds Simulator

A. Objective

i) To expose student with producing digital logic circuit, generating truth table and Timing Diagram with Deeds Simulator

ii) To expose student with a complete cycle process of a combinatorial circuit design and simulate with Deeds Simulator

B. Material

Install Deeds Software for Windows

C. Introduction

Deeds Simulator

The Digital Circuit Simulator ***d-DcS*** appears to the user as a graphical schematic editor, with a library of simplified logic components, specialized toward pedagogical needs and not describing specific commercial products.

As described before, the schematic editor allows building a simple digital networks composed of gates, flip-flops, pre-defined combinational and sequential circuits and custom-defined components (defined as Finite state machine).

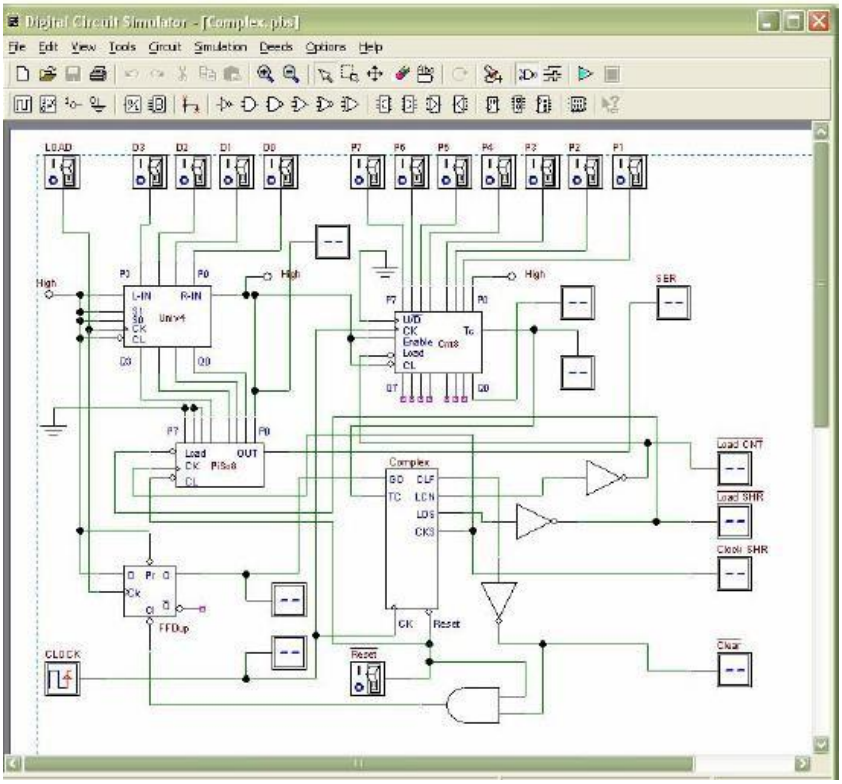


Fig. 1 Circuit Editor of Digital Circuit Simulator (d-DcS)

Simulation can be interactive or in timing-mode. In the first mode, the student can *"animate"* the digital system in the editor, controlling its inputs and observing the results. This is the simplest mode to examine a digital network, and this way of operation can be useful for the beginners. In the timing mode, the behavior of the circuit can be analyzed by a timing diagram window, in which the user can define graphically an input signal sequence and observe the simulation results.

Digital Circuit Simulator (d-DcS): A Simple Example

In following screen shots (Fig. 2a, 2b, and 2c), student can see the circuit during the drawing and then simulated by animation by following this simple steps:

a) student picks-up components from the bin on the Component Tool Bar.

b) connects them using Wires.

c) student activates the animation.

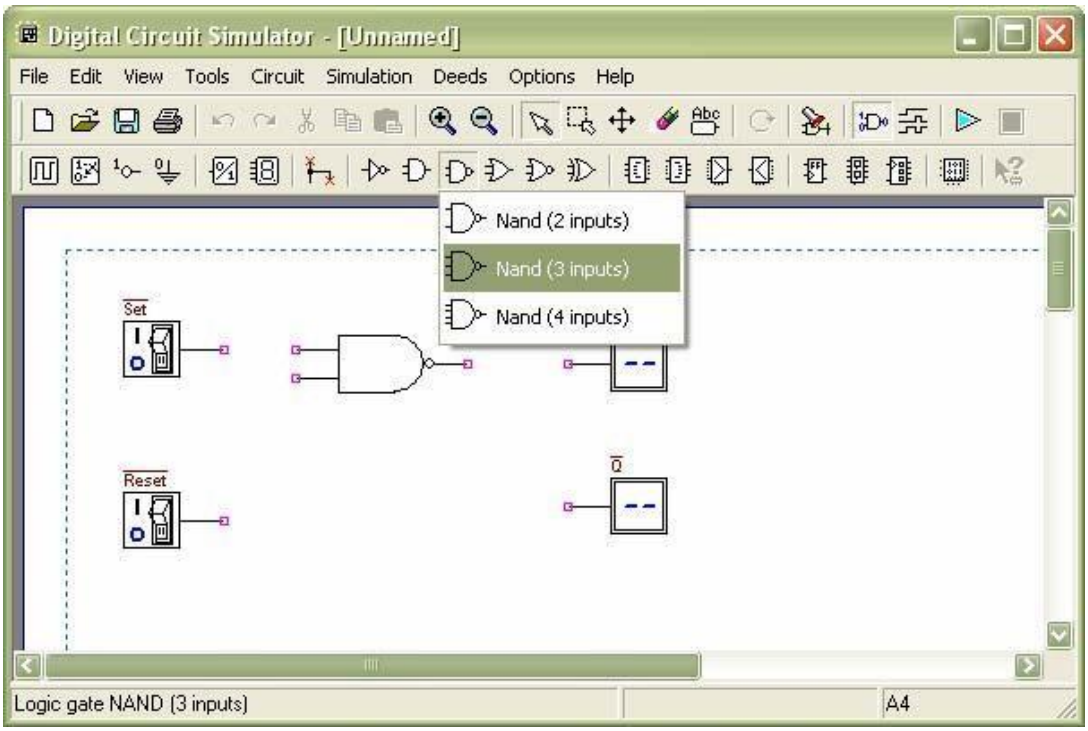


Fig. 2a Drawing Phase of the Digital Circuit Editor: Insertion of Components

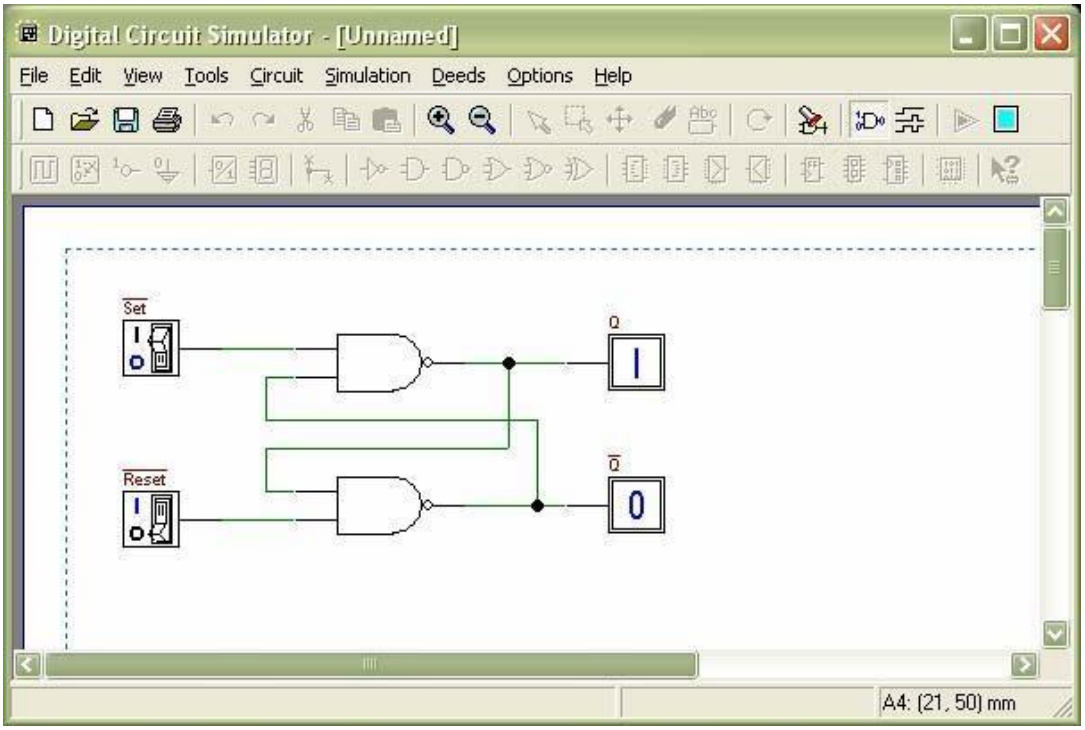


Fig. 2b Next Phase of the Work: Connection of Components using Wires

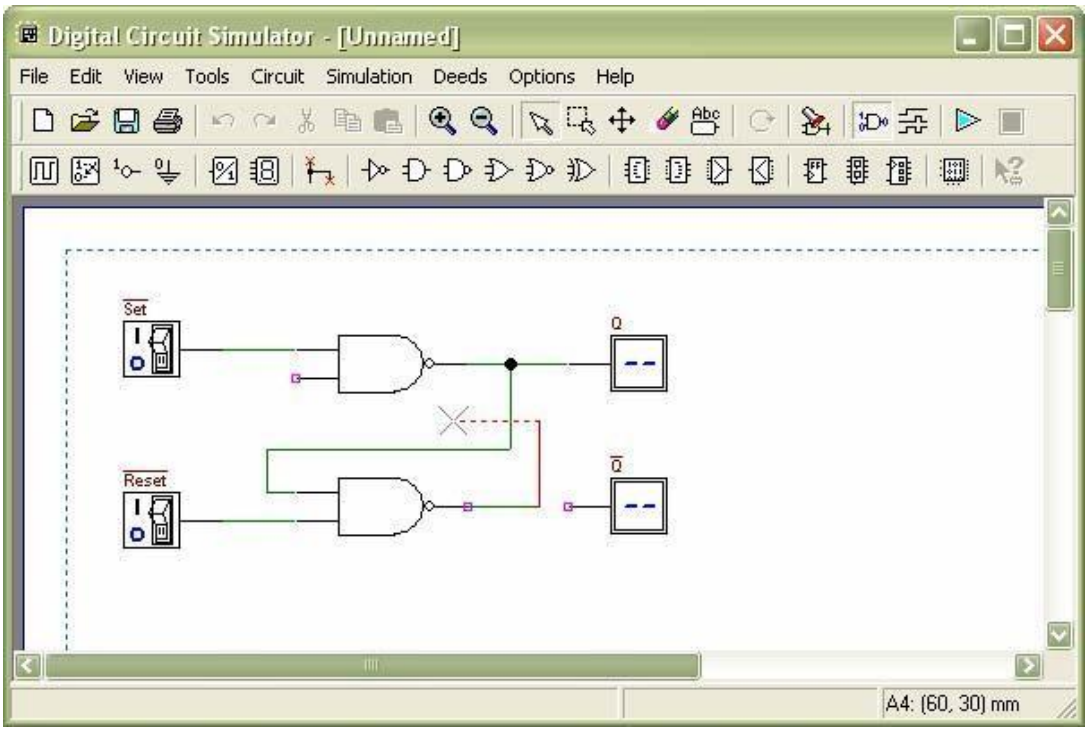


Fig. 2c Animation: User Switches Inputs and the Circuit Shows Changes on Outputs

To exit the ‘animation’ mode, it is necessary to click on the square ‘stop’ button.

Instead, if the timing simulation is to be performed, student should click on the Timing Simulation button. This will show the Timing Diagram simulation window (Fig. 3).

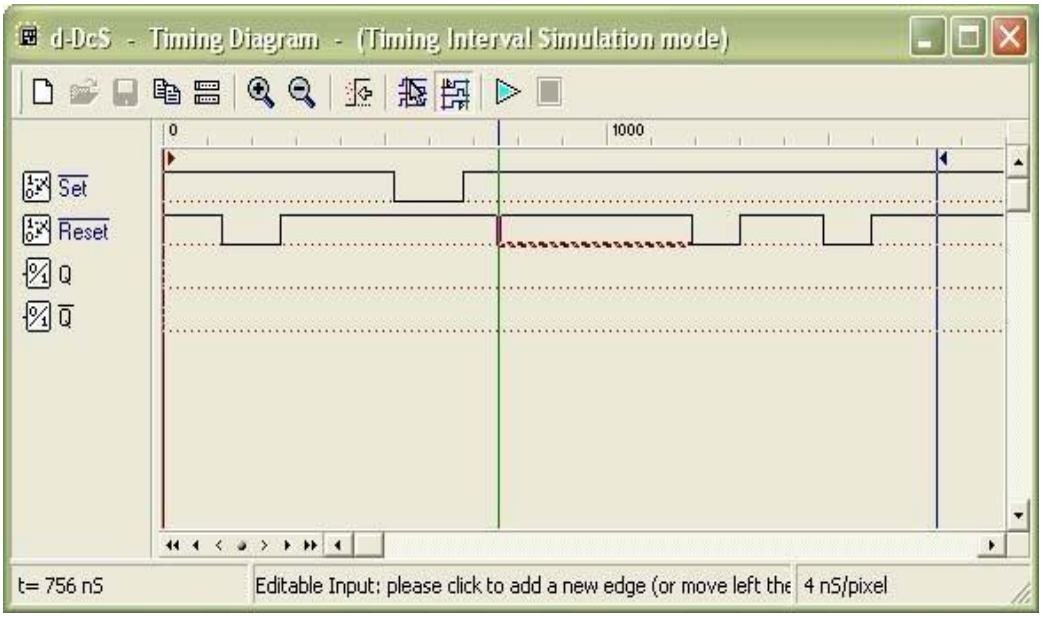


Fig. 3 Timing Diagram Simulation Window

In this window, first student should define the timing of the input signals, drawing them on the diagram with the mouse. A vertical line cursor permits to define the ‘end time’ of the simulation. When student clicks on the triangular ‘play’ button on the toolbar, the simulation is executed, and its results are displayed in the same window (Fig. 4).

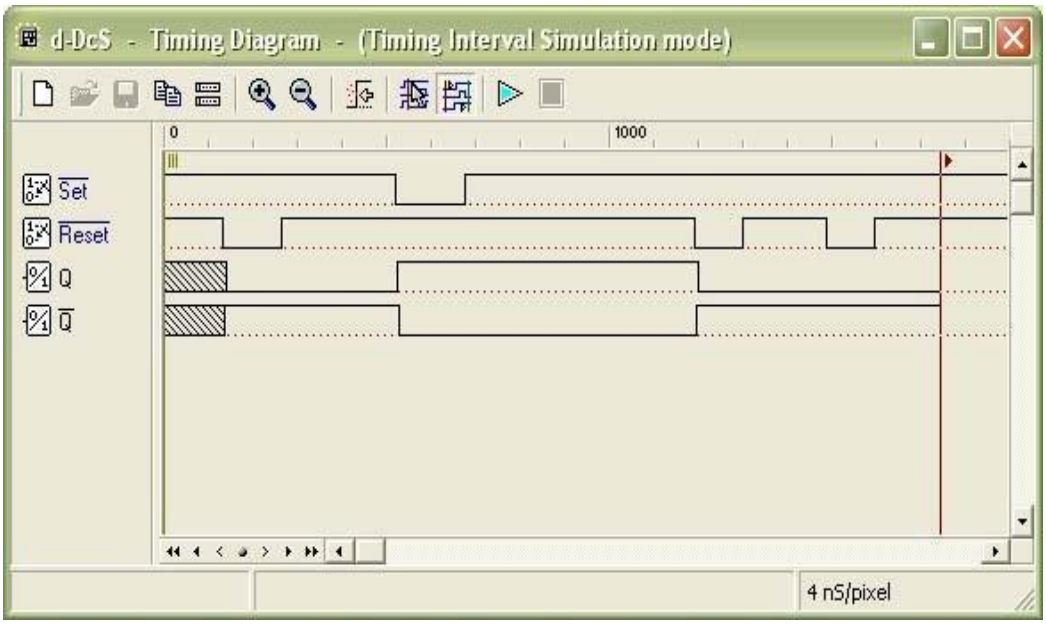


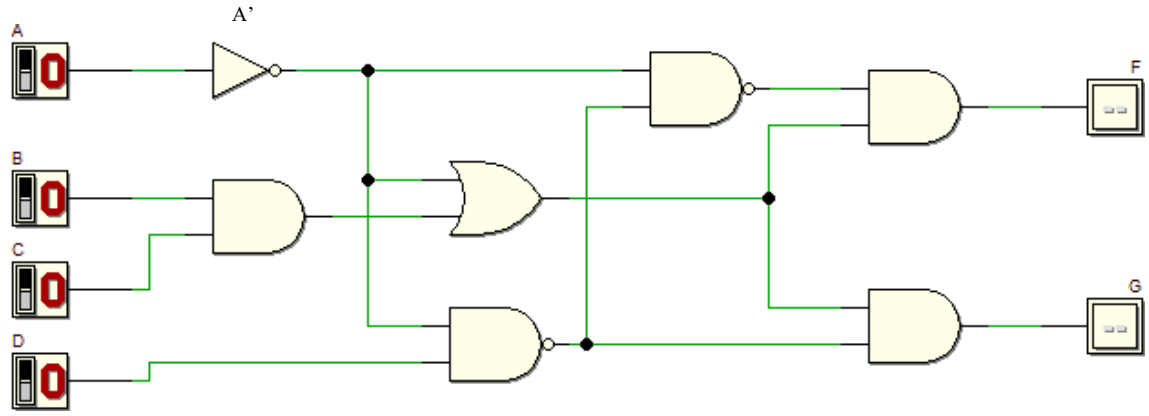
Fig. 4 Timing Simulation Results, Displayed in Timing Diagram Window

Student can verify the correct behavior of the network under test, comparing simulation results with reasoning and theory concepts.

5. Experiment

5.1 Part A:

Refer to circuit in Figure 1.



a) Refer to the circuit. Derive Boolean expression for output F and G.

Output F : [(A' *(A'D)')'] * [A+(BC)]

Output G: [A'+(BC)] * [(A'D)']

b) Simplify equation output F using laws, rules and De Morgan Theorem, and write the equation in Sum of Product (SOP).

[(A' *(A'*D)')']*[A+(B*C)]

= [A''+(A'D)'']*[A'+(BC)] (De Morgan Theorem)

= [A+ (A'D)] * [A'+(BC)] (law 9)

= [(A+A')(A+D)] * [A'+(BC)] (distributive law)

= [1(A+D)] * [A'+(BC)] (law 6)

= AA' + ABC + A'D+ BCD (distributive law)

= 0 + ABC + A'D+ BCD (law 6)

= ABC(D+D')+A'D(B+B')(C+C')+(A+A')BCD

=ABCD+ABCD'+(A'BD+A'B'D)(C+C')+ABCD+A'BCD

=~~ABCD~~+ABCD'+~~A'BCD~~+A'B'CD+A'BC'D+A'B'C'D+ABCD+A'BCD

=ABCD+ABCD'+A'BCD+A'B'CD+A'BC'D+A'B'C'D

c) Simplify equation output G using laws, rules and De Morgan Theorem, and write the equation in Product of Sum (POS).

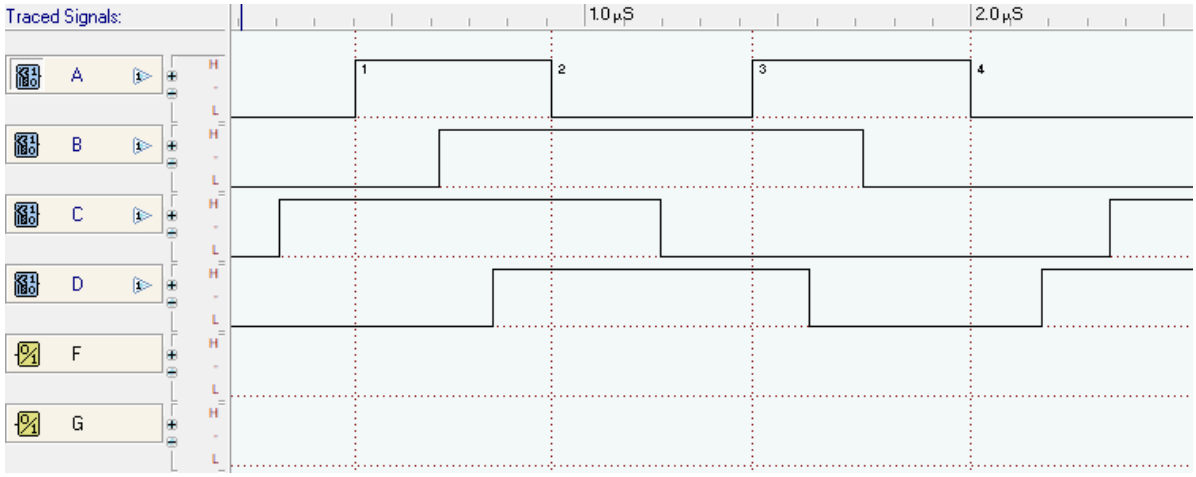
[A'+(BC)] * [(A'*D)']
= [A'+(BC)] * (A''+D') (De Morgan Theorem)
= [A'+(BC)] * (A+D') (law 9)
= (A'+B)(A'+C)(A+D') (Distributive law)
= [A'+B+CC'+DD'][A'+C+BB'+DD'][A+D'+BB'+CC']
= (A'+B+C+DD') (A'+B+C'+DD')
(A'+B+C+DD') (A'+B'+C+DD')
(A+B+D'+CC') (A+B'+D'+CC')
= ~~(A'+B+C+D)~~ ~~(A'+B+C+D')~~ (A'+B+C'+D) (A'+B+C'+D')
(A'+B+C+D) ~~(A'+B+C+D')~~ (A'+B'+C+D) (A'+B'+C+D')
(A+B+C+D') (A+B+C'+D') (A+B'+C+D') (A+B'+C'+D')
= (A'+B+C'+D) (A'+B+C'+D') (A'+B+C+D) (A'+B+C+D') (A'+B'+C+D) (A'+B'+C+D')
(A+B+C+D') (A+B+C'+D') (A+B'+C+D') (A+B'+C'+D')

c) Simulate the circuit and construct the truth table and complete the following.

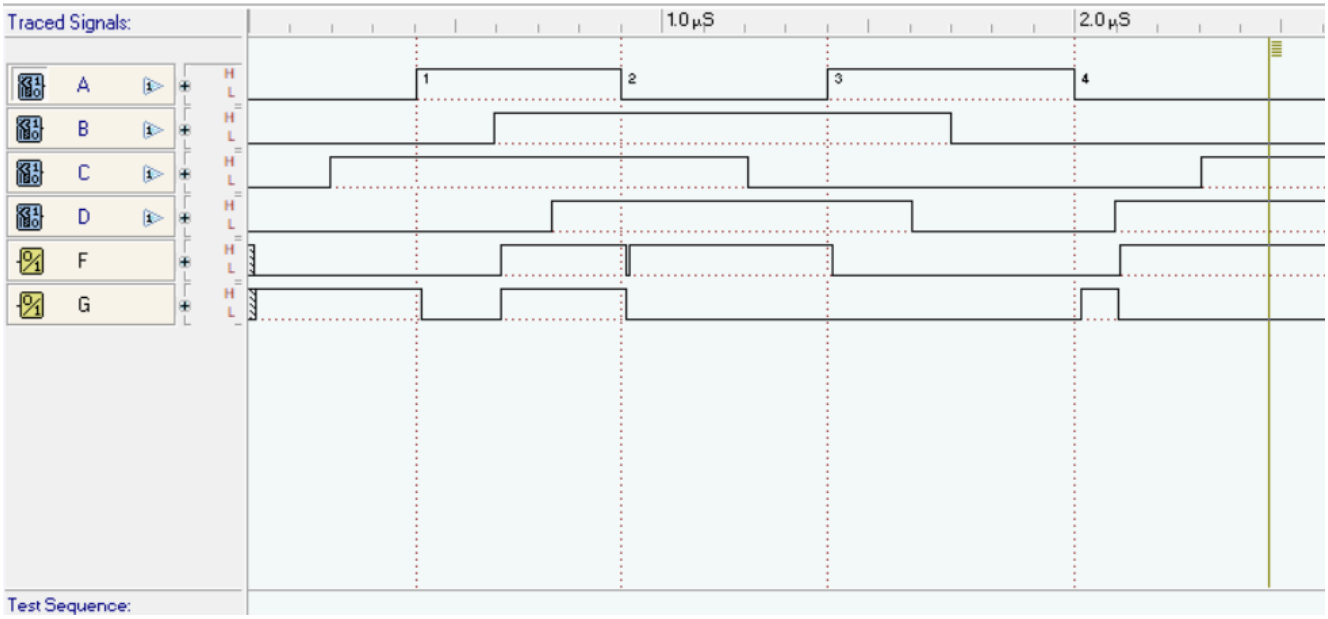
Truth table for output circuit F and G shown in Figure 1

Input				Output	
A	B	C	D	F	G
0	0	0	0	0	1
0	0	0	1	1	0
0	0	1	0	0	1
0	0	1	1	1	0
0	1	0	0	0	1
0	1	0	1	1	0
0	1	1	0	0	1
0	1	1	1	1	0
1	0	0	0	0	0
1	0	0	1	0	0
1	0	1	0	0	0
1	0	1	1	0	0
1	1	0	0	0	0
1	1	0	1	0	0
1	1	1	0	1	1
1	1	1	1	1	1

d) Using Deeds, draw circuit in Figure 1. Simulate and complete the waveform output F and G by referring the following diagram.



Answer



e) Write Boolean equation for output F using Sigma notation.

$F = \sum_{ABCD} (1,3,5,7,14,15)$

f) Write Boolean equation for output G using Pi notation.

$G = \Pi_{ABCD} (1,3,5,7,8,9,10,11,12,13)$

5.2 Part B:

Combinational circuit design process and simulate with Deeds Simulator

Design Process

i) Determine Parameter Input / Output and their relations
ii) Construct Truth Table
iii) Using K-Map, get the SOP optimized form of all Boolean equation outputs
iv) Draw the circuit and use duality symbol; convert AND-OR circuit to NAND gates ONLY.
v) Simulate the design using Deeds Simulator. Check the results according to Truth Table and Timing Diagram Operation.

Problem Situation

Using universal gate **NAND gates** only, design a logic circuit that controls an LRT coach door *OPEN* operation at 3 LRT Stations: *Pandan (S1), Pudu (S2)* and *Maluri (S3).*

Consider the following **inputs**:

- *LRT coach moving status (S)*
  - *S = bit 1 indicates the coach has stopped.*
  - *S = bit 0 indicates the coach is moving.*
- *Sensor location at Station S1, S2, and S3*
  - *HIGH indicates the LRT coach has arrived at the particular station. For instance, S1 = 1 indicates LRT coach has arrived at station S1, and sensor S2=S3=0.*
  - *It is IMPOSSIBLE for the LRT coach to arrive at more than one station at one time.*

The circuit **outputs** are the *OPEN* and *ALARM* signal

- *OPEN = bit 1 indicates the coach door will be opened*
- *ALARM =bit 1 indicates the alarm is activated*

The following are the **condition**s for *OPEN* and *ALARM* outputs at Station *S1*, *S2*, and *S3*

- *The door will be opened ONLY IF the coach stopped at any ONE of the stations.*
- *The alarm will be activated:*
  - *If the coach arrives at any station and it does not stop.* ***or***
  - *If the coach stopped but NOT at stations S1, S2 or S3.*

Experimental steps

i) Construct Truth Table in Table 1 for the LRT operations. Use variables *S, S1, S2,* and *S3* as INPUTS and *OPEN* and *ALARM* as OUTPUTS.

Table 1

INPUT				**OUTPUT**	
S	**S1**	**S2**	**S3**	**OPEN**	**ALARM**
0	0	0	0	0	0
0	0	0	1	0	1
0	0	1	0	0	1
0	0	1	1	x	x
0	1	0	0	0	1
0	1	0	1	x	x
0	1	1	0	x	x
0	1	1	1	x	x
1	0	0	0	0	1
1	0	0	1	1	0
1	0	1	0	1	0
1	0	1	1	x	x
1	1	0	0	1	0
1	1	0	1	x	x
1	1	1	0	x	x
1	1	1	1	x	x

ii) Use K-Map to get optimized SOP Boolean equations for the *OPEN* and *ALARM* circuits.

S2 S3 / S S1	00	01	11	10
00	0	0	x	0
01	0	x	x	x
11	1	x	x	x
10	0	1	x	1

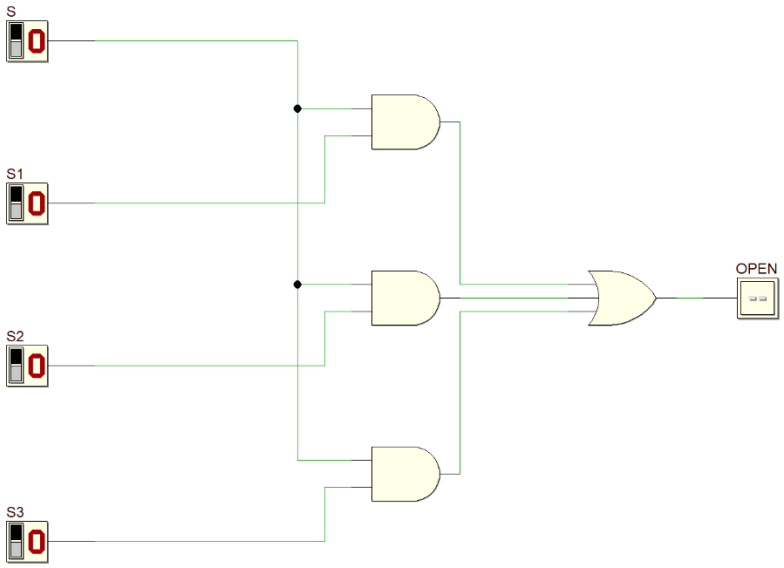
Open = SS1 + SS2 + SS3

S2S3 / SS1	00	01	11	10
00	0	1	x	1
01	1	x	x	x
11	0	x	x	x
10	1	0	x	0

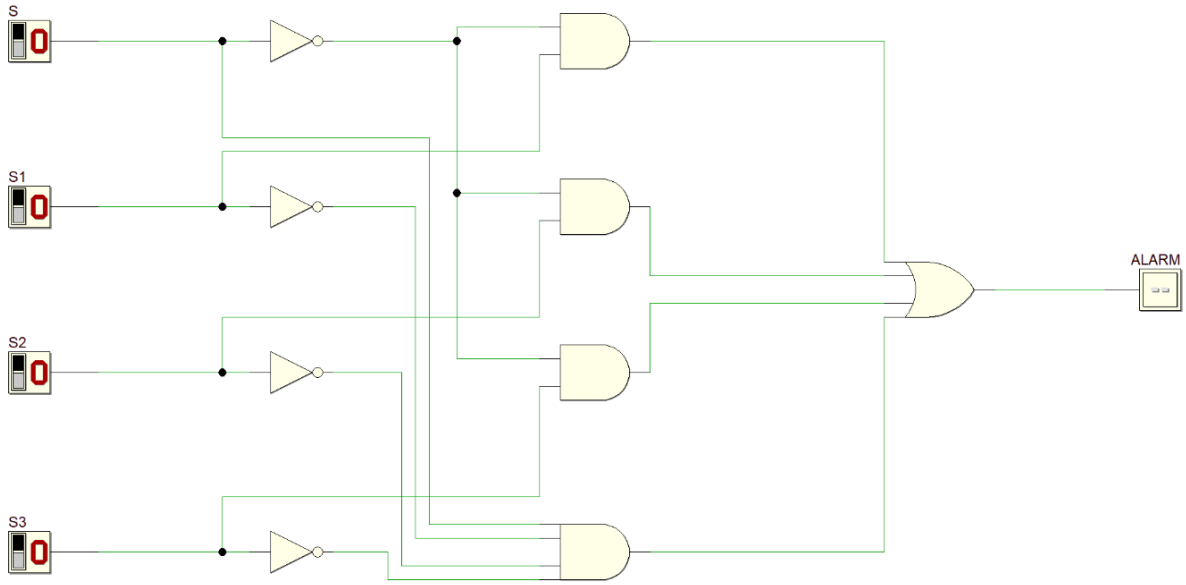
Alarm = S'S1 + S'S2 + S'S3 + SS1'S2'S3'

iii) From equations in (ii), draw your final *OPEN* and *ALARM* circuits using Deeds Simulator.

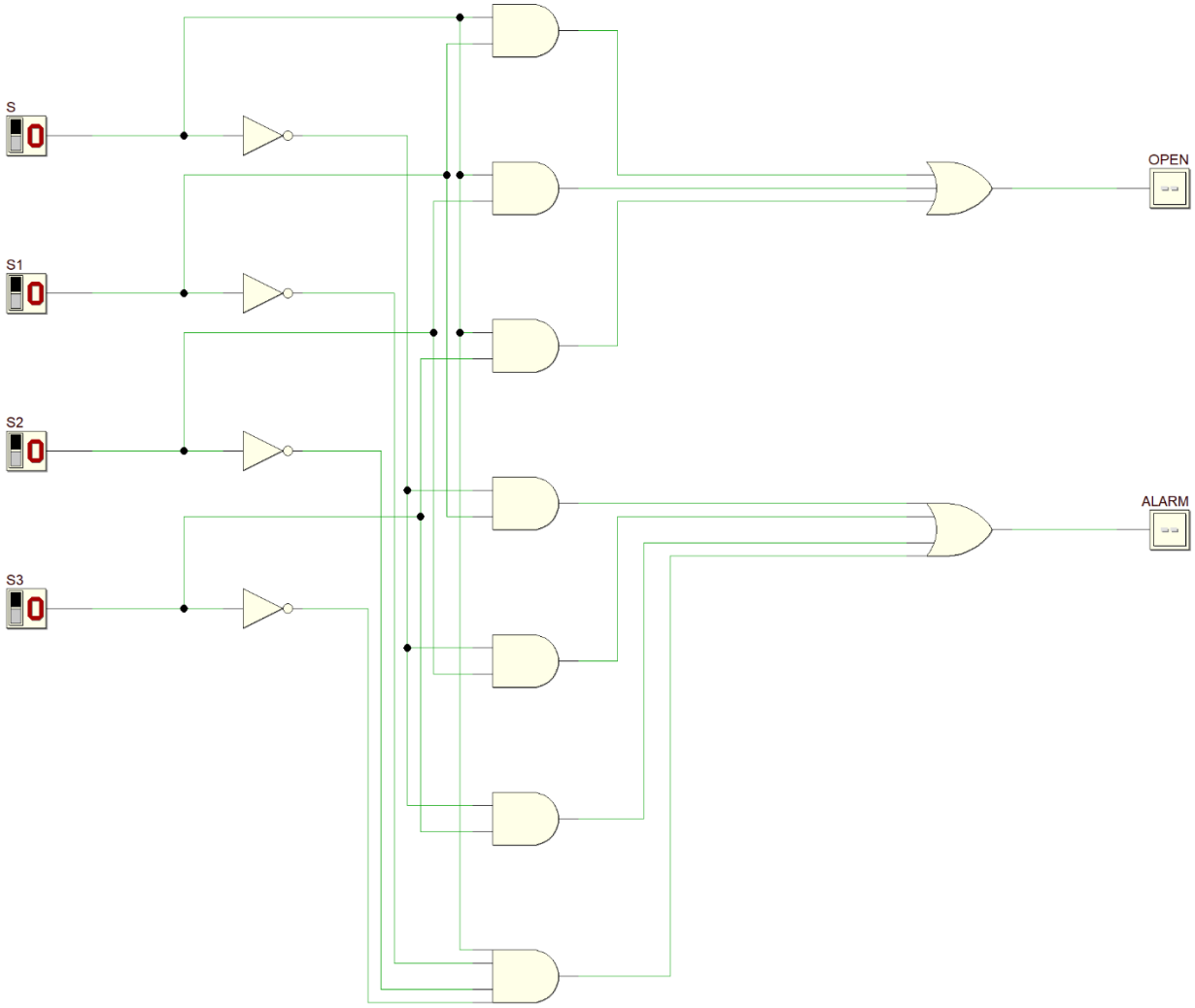
OPEN circuit



ALARM circuit



Combination of OPEN circuit and ALARM circuit



iv) Simulate the circuit design in (iii) and construct Truth Table in Table 2.

Table 2

INPUT				OUTPUT	
S	**S1**	**S2**	**S3**	**OPEN**	**ALARM**
0	0	0	0	0	0
0	0	0	1	0	1
0	0	1	0	0	1
0	0	1	1	0	1
0	1	0	0	0	1
0	1	0	1	0	1
0	1	1	0	0	1
0	1	1	1	0	1
1	0	0	0	0	1
1	0	0	1	1	0
1	0	1	0	1	0
1	0	1	1	1	0
1	1	0	0	1	0
1	1	0	1	1	0
1	1	1	0	1	0
1	1	1	1	1	0

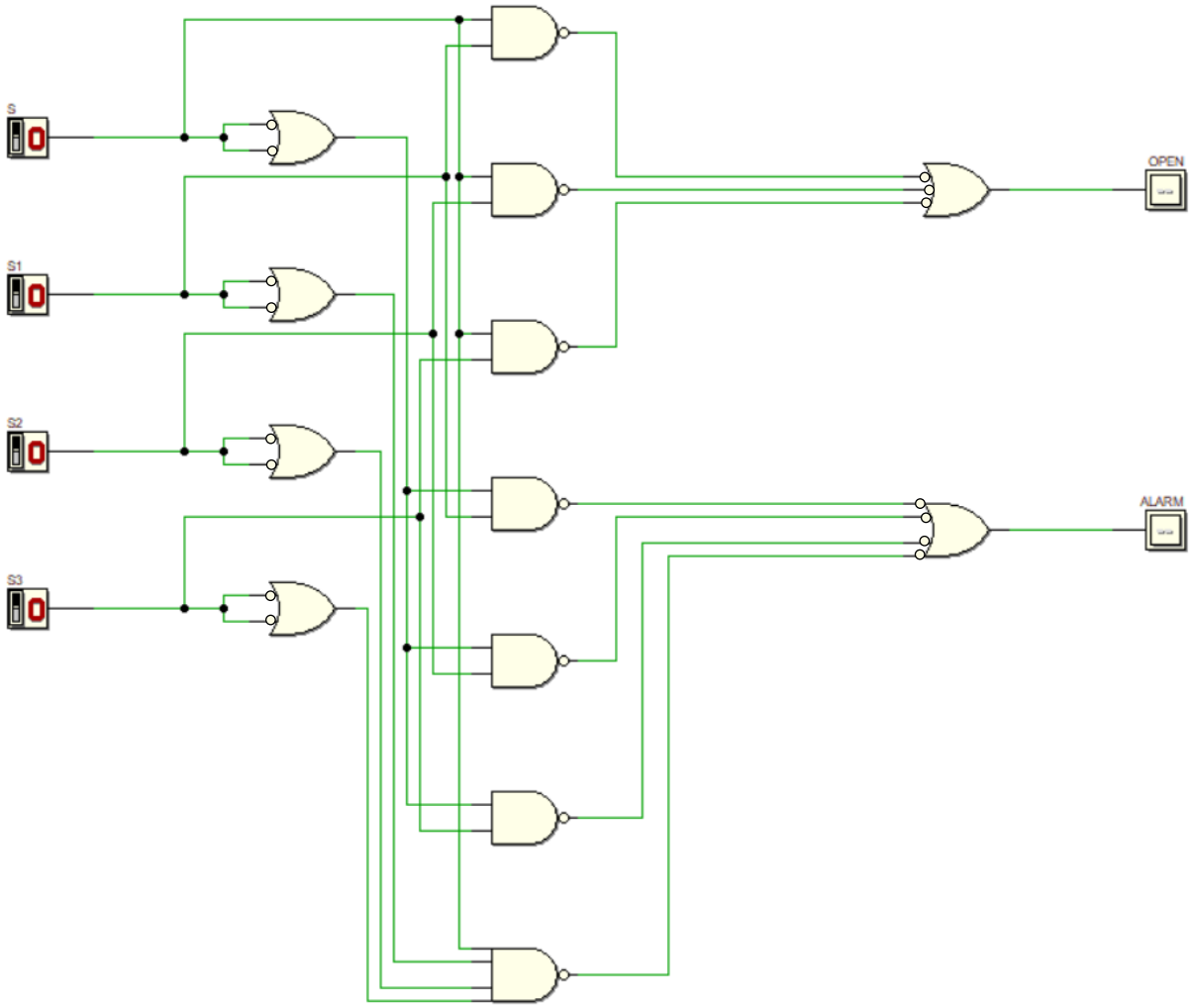
Compare the answer of Table 2 and Table 1. What is your conclusion?

Table 1 used the don't care term, the x represents the output is not exist because the couch cannot reach *Pandan (S1)*, *Pudu (S2)* and *Maluri (S3)* at same time in reality. Table 2 doesn't used the don't care term because the output is for the imaginary situation.

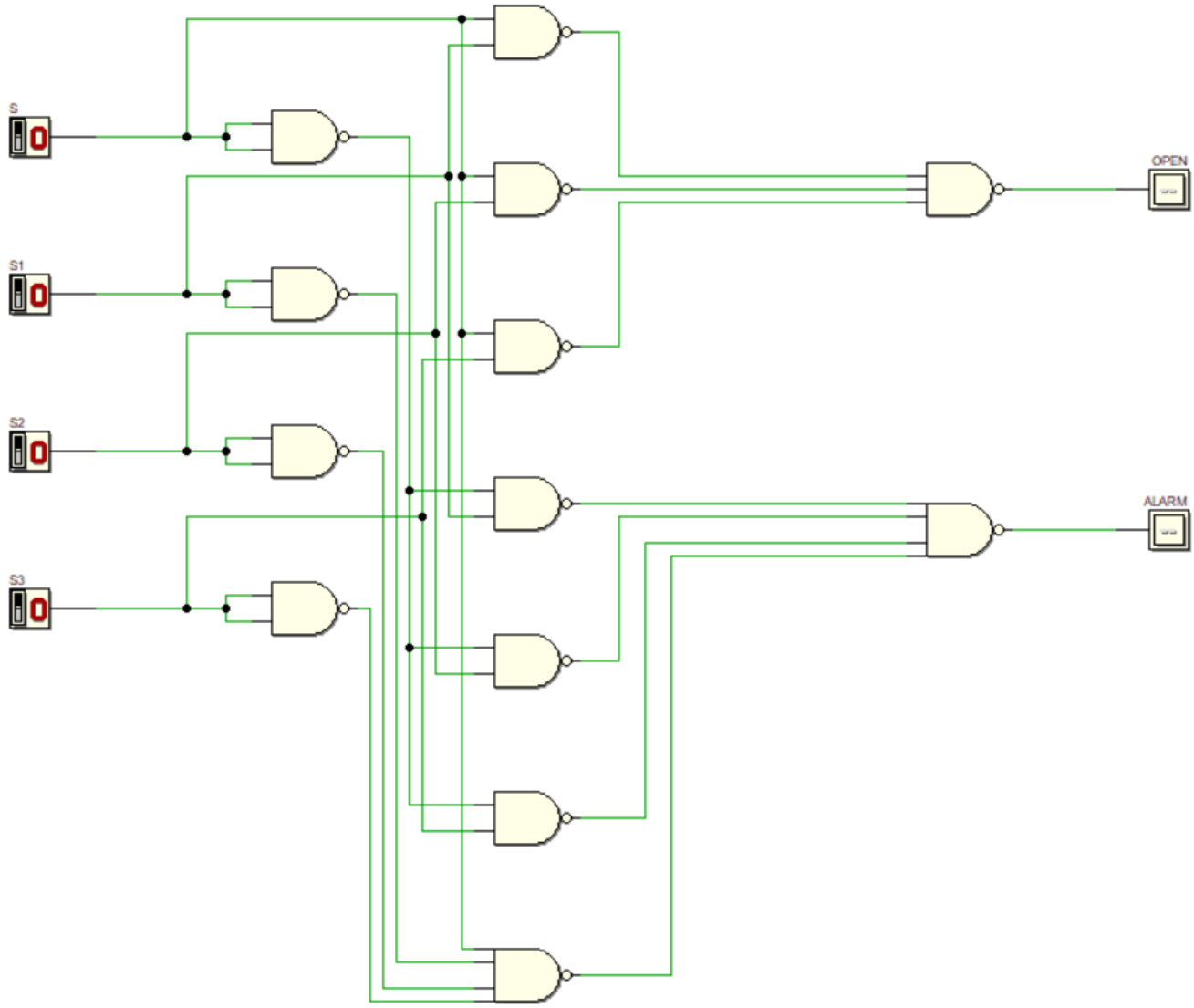
The don't care term is used to simplify the grouping in the K-map to get an expression.

v) Use dual symbol to convert, AND-OR circuit to NAND gates only. Draw the final circuit using Deeds Simulator.

Dual Symbol (Negative -OR)



NAND gate



vi) Simulate the final NAND gates design in (v) and construct Truth Table in Table 3.

Table 3

INPUT				OUTPUT	
S	**S1**	**S2**	**S3**	**OPEN**	**ALARM**
0	0	0	0	0	0
0	0	0	1	0	1
0	0	1	0	0	1
0	0	1	1	0	1
0	1	0	0	0	1
0	1	0	1	0	1
0	1	1	0	0	1
0	1	1	1	0	1
1	0	0	0	0	1
1	0	0	1	1	0
1	0	1	0	1	0
1	0	1	1	1	0
1	1	0	0	1	0
1	1	0	1	1	0
1	1	1	0	1	0
1	1	1	1	1	0

Compare the answer of Table 2 and Table 3. What is your conclusion?

Table 3 and Table 4 are identical. The conclusion, the NAND gate is a universal gate that NAND gate can represent the AND and OR gate by using the particular amount of NAND gate.

Fully Completed ☐ Partially Completed ☐ Checked by: _____