CS 312: Computer Architecture II

Name: Orchlon Chinbat
Student ID: 50291063

Honor Code: I pledge that this submission is all my own work and not a copy nor a partial copy from anywhere.

Lab 2. Logic Gates II

Use the MultimediaLogic (MMLogic) system for the simulations.

Part 1.

a) Complete Problem 12.8 at the end of Chapter 12 Digital Logic in the textbook.

a. Truth Table:

$X_1X_2X_3X_4$	Decimal	Z_1	Z_2	Z_3	Z_4	Z_5	Z_6	Z_7
0000	0	1	1	1	0	1	1	1
0001	1	0	0	1	0	0	1	0
0010	2	1	0	1	1	1	0	1
0011	3	1	0	1	1	0	1	1
0100	4	0	1	1	1	0	1	0
0101	5	1	1	0	1	0	1	1
0110	6	1	1	0	1	1	1	1
0111	7	1	0	1	0	0	1	0
1000	8	1	1	1	1	1	1	1
1001	9	1	1	1	1	0	1	0

b. SOP Form (Sum of Product):

Minterms:

0. 0000: $X_1'X_2'X_3'X_4'$

1. 0001: $X_1'X_2'X_3'X_4$

2. 0010: $X_1'X_2'X_3X_4'$

3. 0011: $X_1'X_2'X_3X_4$

4. 0100: $X_1'X_2X_3'X_4'$

5. 0101: $X_1'X_2X_3'X_4$

6. 0110: $X_1'X_2X_3X_4'$

7. 0111: $X_1'X_2X_3X_4$

8. 1000: $X_1X_2'X_3'X_4$

9. 1001: $X_1X_2'X_3'X_4$

The sum-of-products expression simplified using minterms:

Explanation: This represents a straightforward sum notation using 'm' for minterms, where all the indices inside the parentheses are added together.

$Z_1 = \sum m(0, 2, 3, 5, 6, 7, 8, 9)$

$Z_2 = \sum m(0, 4, 5, 6, 8, 9)$

$Z_3 = \sum m(0, 1, 2, 3, 4, 7, 8, 9)$

$Z_4 = \sum m(2, 3, 4, 5, 6, 8, 9)$

$Z_5 = \sum m(0, 2, 6, 8)$

$Z_6 = \sum m(0, 1, 3, 4, 5, 6, 7, 8, 9)$

$Z_7 = \sum m(0, 2, 3, 5, 6, 8, 9)$

c. POS:
Maxterms:

0. 0000: $X_1+X_2+X_3+X_4$

1. 0001: $X_1+X_2+X_3+X_4'$

2. 0010: $X_1+X_2+X_3'X_4$

3. 0011: $X_1+X_2+X_3'+X_4'$

4. 0100: $X_1+X_2'+X_3+X_4$

5. 0101: $X_1+X_2+X_3+X_4'$

6. 0110: $X_1+X_2'+X_3'+X_4$

7. 0111: $X_1'+X_2'+X_3'+X_4'$

8. 1000: $X_1'+X_2+X_3+X_4$

9. 1001: $X_1'+X_2+X_3+X_4'$

The Product of Sums (POS) expression simplified using maxterms:

Explanation: This uses a simple product notation with 'M' to represent maxterms, indicating that all the indices within the parentheses are combined through multiplication.

$Z_1 = \prod M(1, 4)$

$Z_2 = \prod M(1, 2, 3, 7)$

$Z_3 = \prod M(5, 6)$

$Z_4 = \prod M(0, 1, 7)$

$Z_5 = \prod M(1, 3, 4, 5, 7, 9)$

$Z_6 = \prod M(2)$

$Z_7 = \prod M(1, 4, 7)$

d. Simplified expression:

$Z_1 = X_1 + X_3 + X_2X_4 + X_2'X_4'$

$Z_2 = X_1 + X_2X_3' + X_2X_4' + X_3'X_4'$

$Z_3 = X_2' + X_3X_4 + X_3'X_4'$

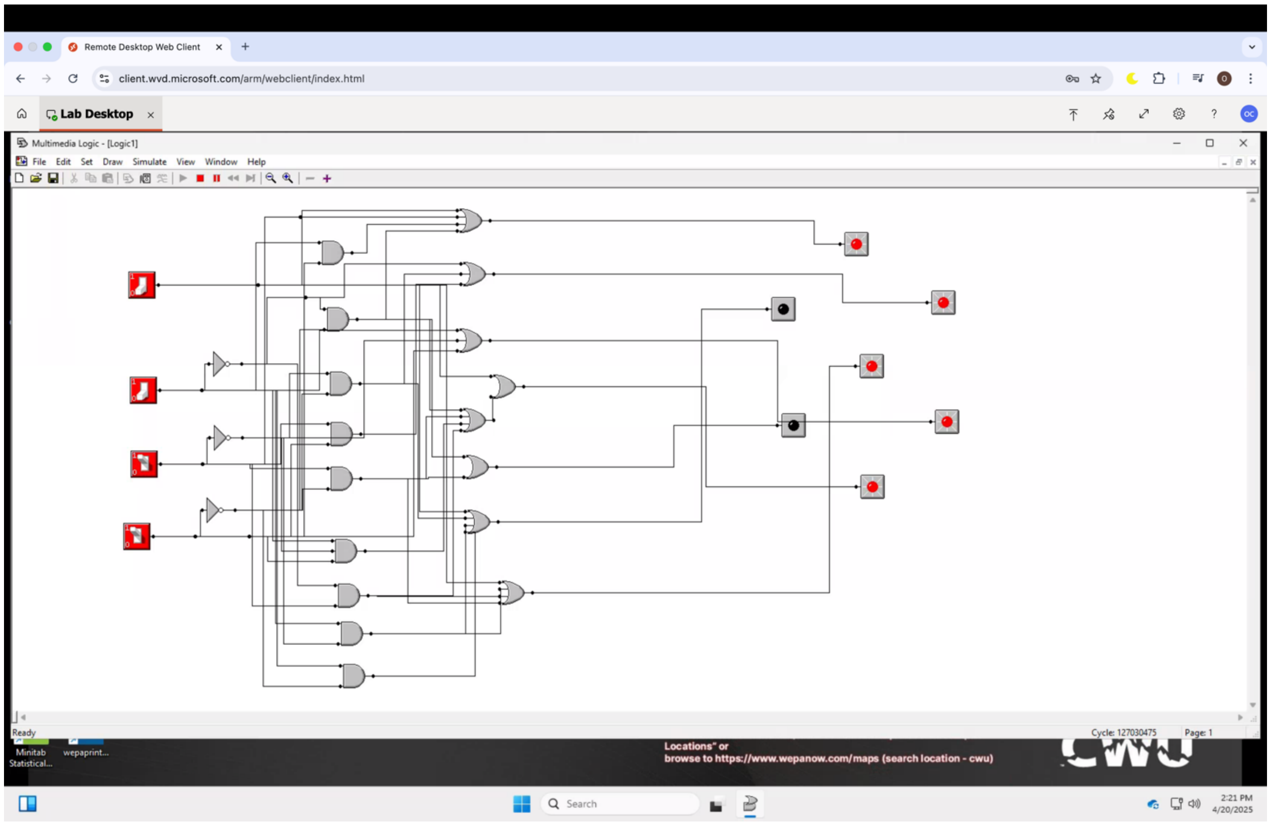
$Z_4 = X_1 + X_2X_3' + X_2'X_3 + X_3X_4'$

$Z_5 = X_3X_4' + X_2'X_4'$

$Z_6 = X_2 + X_4 + X_3'$

$Z_7 = X_1 + X_3X_2' + X_3X_4' + X_2'X_4' + X_2X_4X_3'$

b). Implement the seven-segment display using the basic logic gates and simulate your implementations using MMlogic – Do not use the ready-to-use 7-segment-LED; you may use the LED to form the 7 segments. Paste a screenshot to show your implementation. Explain your results.


Remote Desktop Web Client
client.wvd.microsoft.com/arm/webclient/index.html
Lab Desktop
Multimedia Logic - [Logic1]
File Edit Set Draw Simulate View Window Help
Ready
Cycle: 127030475
Page: 1
Minitab Statistical...
wepaprint...
Locations" or
browse to https://www.wepanow.com/maps (search location - cwu)
Search
2:21 PM
4/20/2025

Part 2.

a) Complete Problem 12.13 at the end of Chapter 12 Digital Logic in the textbook. Explain your solution.

Sum (S) Calculation:

- First XOR gate (XOR1): Compute A⊕B.
- Second XOR gate (XOR2): Compute (A⊕B)⊕C. This gives the sum **S**.

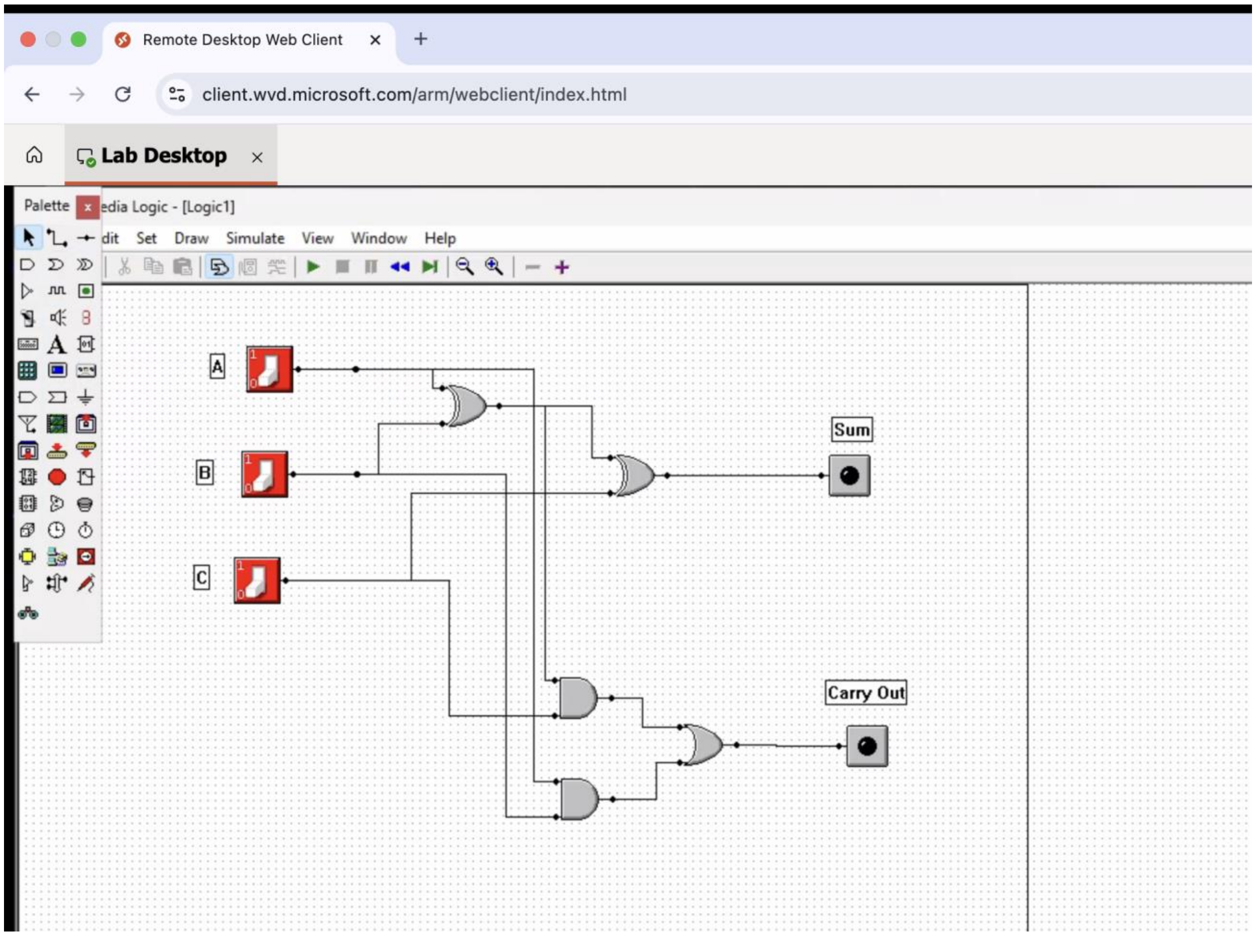
Carry-out (C_{out}) Calculation:

- Third AND gate (AND1): Compute AB.
- Fourth gate (AND2): Compute (A⊕B)C.
- Fifth gate (OR1): Combine the results of AND1 and AND2 to get

  $\mathbf{C_{out}}$=(AB) + (C(A⊕B)).

The full adder can be implemented with five gates: two XORs, two ANDs, and one OR. The sum is **S** = (A⊕B) ⊕ C, and the carry-out is $\mathbf{C_{out}}$ = (AB) + (C(A⊕B)).

b) Simulate your implementation using MMLogic and paste a screenshot of your circuit. Explain your results.



Part 3.

a) Complete Problem 12.16 at the end of Chapter 12 Digital Logic in the textbook. Explain your solution.

S-R flip-flop to D flip-flop:

To make the S-R flip-flop behave like a D flip-flop, we need to connect the S and R inputs to a new D input in a way that:

- When D=1, the flip-flop sets (Q=1), which requires S=1 and R=0.
- When D=0, the flip-flop resets (Q=0), which requires S=0 and R=1.

This can be achieved by:

- Connecting the D input directly to S.
- Connecting the D input through an **inverter** to R, so R becomes the opposite of D (R = not D).

Here's why it works:

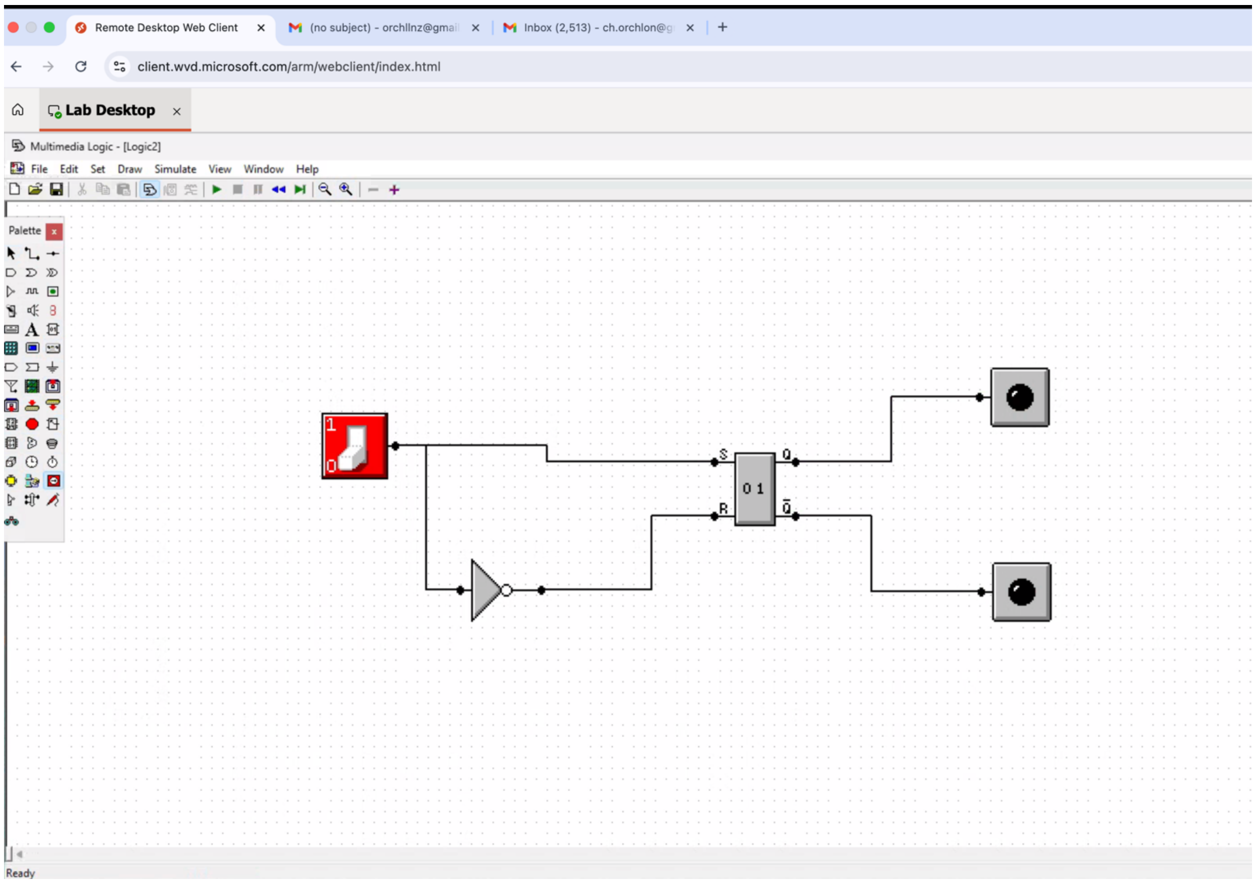
- When D = 1:
  - S = D = 1
  - R = not D = 0
  - This sets Q to 1.
- When D = 0:
  - S = D = 0
  - R = not D = 1
  - This resets Q to 0.

Since R is always the inverse of D, the invalid state (S=1 and R=1) never occurs.

Modifications to the Graphic Symbol

1. **Add a new input labeled D**: This represents the data input of the D flip-flop.
2. **Draw a line from D to S**: Connect the D input directly to the S input of the S-R flip-flop.
3. **Add an inverter**: Place an inverter (NOT gate) with its input connected to D.
4. **Draw a line from the inverter's output to R**: Connect the output of the inverter (which is not D) to the R input of the S-R flip-flop.

b) Simulate your implementation using MMLogic and paste a screenshot of your circuit. Explain your results.


Remote Desktop Web Client
(no subject) - orchllnz@gmail
Inbox (2,513) - ch.orchlon@g
client.wvd.microsoft.com/arm/webclient/index.html
Lab Desktop
Multimedia Logic - [Logic2]
File Edit Set Draw Simulate View Window Help
Palette
1
0
S
R
Q
0 1
Ready

Grading rubric:

Parts	Points
Part 1.a	8
Part 1.b	7
Part 2.a	5
Part 2.b	5
Part 3.a	5
Part 3.b	5

Submission: This file must be submitted through Canvas.