NTUST OOP Midterm Problem Design

Subject: Simulating Propagation Delays in Digital Logic Gates

Contributor: 沈登獻

Main testing concept:

Design and implementation using classes, objects, constructors, and string handling to manage dynamic behaviors and user commands.

Basics

■ C++ BASICS
■ FLOW OF CONTROL
■ FUNCTION BASICS
□ PARAMETERS AND OVERLOADING
■ ARRAYS
■ STRUCTURES AND CLASSES
■ CONSTRUCTORS AND OTHER TOOLS
□ OPERATOR OVERLOADING, FRIENDS, AND REFERENCES
■ STRINGS
□ POINTERS AND DYNAMIC ARRAYS

Functions

□ SEPARATE COMPILATION AND NAMESPACES
□ STREAMS AND FILE I/O
■ RECURSION
□ INHERITANCE
□ POLYMORPHISM AND VIRTUAL FUNCTIONS
□ TEMPLATES
□ LINKED DATA STRUCTURES
■ EXCEPTION HANDLING
■ STANDARD TEMPLATE LIBRARY
□ PATTERNS AND UML

Description

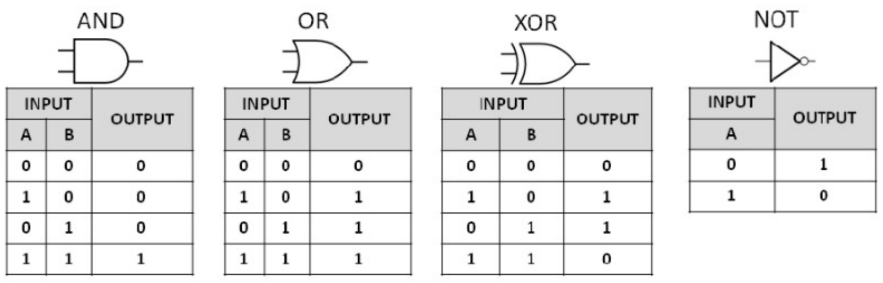
Digital logic circuits are electronic circuits that process digital signals using logic gates. The basic elements of these circuits are binary states, typically represented by 0 (off) and 1 (on). Digital logic circuits are widely used in modern electronic devices, especially in computer hardware and digital system design.

Key Concepts

Logic Gates:

Logic gates are the fundamental building blocks of digital circuits. They determine the output based on the combination of their input signals. The most common logic gates include:

- AND: Outputs 1 only if all inputs are 1; otherwise, outputs 0.
- OR: Outputs 1 if at least one input is 1; otherwise, outputs 0.
- NOT: Inverts the input signal; if the input is 1, the output is 0, and vice versa.
- NAND: The inverse of AND; outputs 0 only if all inputs are 1; otherwise, outputs 1.
- NOR: The inverse of OR; outputs 1 only if all inputs are 0; otherwise, outputs 0.
- XOR: Outputs 1 if and only if the two inputs are different.
- XNOR: The inverse of XOR; outputs 1 if the two inputs are the same; otherwise, outputs 0.



AND

INPUT		OUTPUT
A	B	
0	0	0
1	0	0
0	1	0
1	1	1

OR

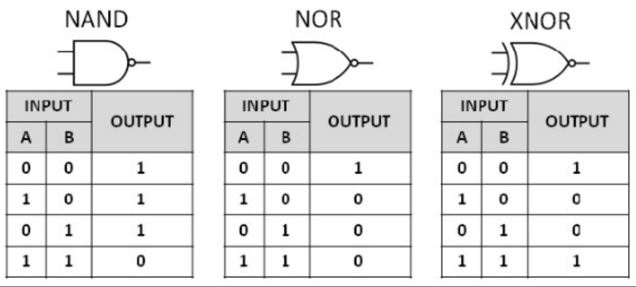
INPUT		OUTPUT
A	B	
0	0	0
1	0	1
0	1	1
1	1	1

XOR

INPUT		OUTPUT
A	B	
0	0	0
1	0	1
0	1	1
1	1	0

NOT

INPUT	OUTPUT
A	
0	1
1	0



NAND

INPUT		OUTPUT
A	B	
0	0	1
1	0	1
0	1	1
1	1	0

NOR

INPUT		OUTPUT
A	B	
0	0	1
1	0	0
0	1	0
1	1	0

XNOR

INPUT		OUTPUT
A	B	
0	0	1
1	0	0
0	1	0
1	1	1

Boolean Algebra:

The operations of digital circuits are based on Boolean algebra, a branch of mathematics dealing with binary variables (0 and 1).

Digital Signals:

Digital circuits handle discrete signals, typically represented in binary form. Each signal can be either True or False(0 and 1).

Propagation Delay:

Propagation delay is the time it takes for a signal to travel from one component to another within the circuit. Each logic gate has its own propagation delay depends on its type.

Programming Task:

You are required to write a program that simulates combinational logic circuits with propagation delays. The simulation will build the circuit by processing various commands that add signals, set their values at specific times, create logic gates, connect them, and adjust their propagation delays. Users can then query the output of any logic gate at a given time.

About Query & Propagation Delay:

If the A and B signals are connected to the AND gate G1:
t = 0.0, A = False, B = False
t = 3.0, A = True
t = 5.0, B = True

Since the delay is 3.0 , G1 will output True at t = 8.0.
The Query command will NOT overlap with the delay time; only the final output should be considered.
In this example, 3.0 <= t <= 8.0, there will be no Query operations.

Commands:

- **Add signal <signal name>**
  Adds a new signal with the specified name to the simulation.
- **Set <signal name> <True/False> <t (ns)>**
  Sets the value of the specified signal (either True or False) at the given time **t**.
- **Add <logic gate type> <component name>**
  Adds a new logic gate to the simulation. The logic gate can be one of the following types: AND, OR, NOT, NAND, NOR, XOR, or XNOR. Each gate component must have a unique name.
  The default propagation delay for any gate before assignment is 3.0 ns.
- **Bind <logic gate output / input signal> <input gate name> <pin id>**
  Connects a signal or the output of one gate to a specified pin (0, 1, etc.) of another gate. Note that all logic gate types have two pins (0 and 1), except the NOT gate, which only has pin 0.
- **Query <logic gate> <t (ns)>**
  Outputs the simulation result at time t for the specified logic gate. The output should be displayed as either True or False based on the computed output at that time.
- **EXIT**
  Terminates the program.

Detail Description :

- Initially, the circuit is empty.
- After that, the program will accept a series of commands that configure the circuit. All **signals** default to a value of False until explicitly set.
- After the circuit is built, a command in the format "Query <logic gate> <t>" will be called to output the status of a specific logic gate at time t (in nanoseconds).

- Only the **Set** and **Query** commands include an explicit time parameter (t); all other commands (such as **Bind**, **Add signal**, and **Add**) are **assumed** to execute at t = 0 and are guaranteed to complete **before any Query operation.**
- The Set command, formatted as “Set <signal name> <True/False> <t>”, immediately appends to the signal’s history. After this command is issued, the simulation **recalculates the logic gate outputs** at Query time t using the updated signal history.

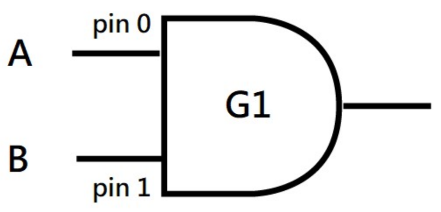
Hint:

- The name of a signal or logic gate is a continuous string without escape characters.
- Each logic gate type is associated with a propagation delay, which affects the timing of when the output signal becomes valid after the input changes.
- The simulation is designed to model combinational logic; thus, the output of a logic gate is solely determined by its current inputs and their propagation delays.
- It is guaranteed that all input data will form a valid circuit configuration.
- **BEFORE** the logic gates' pin0 and pin1 are connected to the signals, please default the input of pin0 / pin1 to False.

Example Workflow:

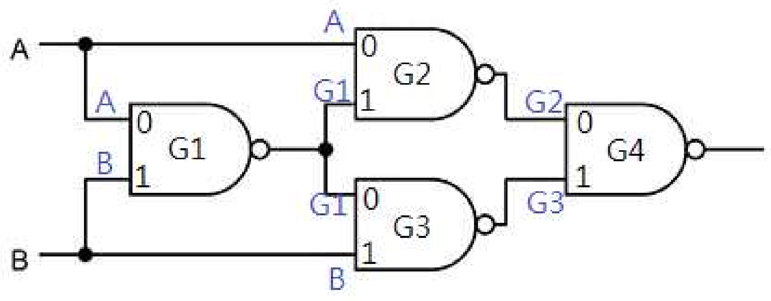
- **Add a signal and set its value:**
  - Add A
    (A signal named "A" is added to the simulation.)
  - Set A True 5.0
    (Signal "A" is set to True at 5.0 ns.)
- **Add a logic gate and set its delay:**
  - Add AND G1
    (An AND gate named "G1" is added.)
- **Connect the signal to the logic gate:**
  - Bind A G1 0
    (Signal "A" is connected to pin 0 of gate "G1".)
  - Bind G1 G2 1
    (The output of gate "G1" is connected to pin 1 of gate "G2". This means that the output of gate "G1" will serve as an input for gate "G2" on pin 1.)
- **Display simulation result:**
  - Query G1 10.0
    (Outputs the computed result at gate "G1" at time 10 ns, considering the propagation delays.)

Sample Input Case 1: AND Gate Simulation



Sample Input	Sample Output
Add signal A Set A True 5.0 Add signal B Set B True 3.0 Add AND G1 Bind A G1 0 Bind B G1 1 Query G1 9.0 Query G1 13.0 EXIT	True True

Sample Input Case 2: XOR Gate Simulation



Sample Input	Sample Output
Add signal A Set A True 0.0 Add NAND G1 Add NAND G2 Add NAND G3 Add NAND G4 Add signal B Set B True 5.0 Bind A G1 0 Bind B G1 1 Bind G1 G2 1 Bind G1 G3 0 Bind A G2 0 Bind B G3 1 Bind G2 G4 0 Bind G3 G4 1 Query G4 15.0 Set B False 9.0 Query G1 15.0 EXIT	False True

□ **Easy, Only basic programming syntax and structure are required.**

□ Medium, Multiple programming grammars, and structures are required. **■ Hard, Need to use multiple program structures or complex data types.**
Expected solving time: 120 minutes
Other Notes: