National Yang Ming Chiao Tung University
Department of Electrical Engineering
Data Structures, Fall 2022

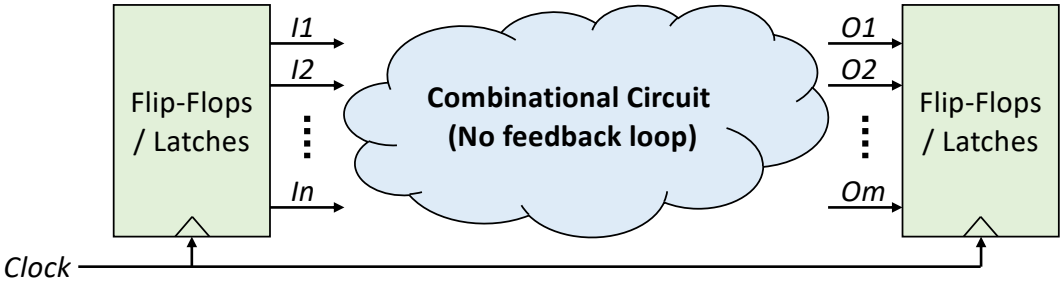
November 30, 2022
Instructor: Mark P.-H. Lin
TA: Chen-Wen Chen

Programming Assignment #5

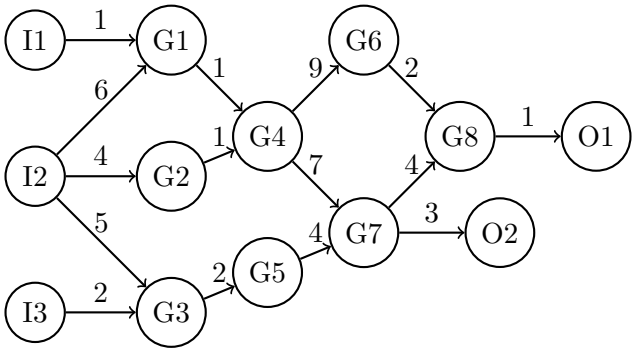
Graphs

1 Problem Description

When designing digital integrated circuits (ICs), such as microprocessors or AI accelerators, the clock frequency/speed, is one of the major performance metrics. *Timing analysis* is a process of verifying whether the design meets its speed requirement, which helps determine the fastest permissible clock speed (e.g. 1 GHz) by calculating signal propagation delay (e.g. 1 ns.) along the longest path of a combinational circuit from register (flip-flops/latches) to register (flip-flops/latches), as shown in the figure below.



In this programming assignment, you are given a logic network represented by a directed acyclic graph (DAC) corresponding to a combinational circuit. The figure below demonstrates an example DAC, which is converted from a combinational circuit.



Each vertex in the DAC denotes either an input (I_i), an output (O_j), or a logic gate (G_k), and each directed edge represents the interconnection from an input to a logic gate, or from a logic gate to another logic gate, or from a logic gate to an output. The edge weight denotes the signal propagation delay of the corresponding interconnection. Based on the given inputs, you are asked to calculate the signal propagation delay along the longest path of a combinational circuit, and identify all the longest paths in the given circuit, as well as the criticality of each interconnection.

2 Input Format

The input file "sample.in" for the aforementioned DAG is given below, which lists all directed edges and the corresponding weights. To simply the problem, the maximum number of directed edges is limited to 50.

sample.in	Comments
I1 G1 1	// Edge:<I1, G1>, Weight: 1
I2 G1 6	// Edge:<I2, G1>, Weight: 6
I2 G2 4	// Edge:<I2, G2>, Weight: 4
I2 G3 5	// Edge:<I2, G3>, Weight: 5
I3 G3 2	// Edge:<I3, G3>, Weight: 2
G1 G4 1	// Edge:<G1, G4>, Weight: 1
G2 G4 1	// Edge:<G2, G4>, Weight: 1
G3 G5 2	// Edge:<G3, G5>, Weight: 2
G4 G6 9	// Edge:<G4, G6>, Weight: 9
G4 G7 7	// Edge:<G4, G7>, Weight: 7
G5 G7 4	// Edge:<G5, G7>, Weight: 4
G6 G8 2	// Edge:<G6, G8>, Weight: 2
G7 G8 4	// Edge:<G7, G8>, Weight: 4
G8 O1 1	// Edge:<G8, O1>, Weight: 1
G7 O2 3	// Edge:<G7, O2>, Weight: 3

3 Output Format

Your program will need to generate an output file which follows the format in "sample.out". Similar to the textbook and lecture note, you will need to output the earliest time, the latest time, the slack, and the criticality of each directed edge. Moreover, the maximum latency of the critical paths, and all the critical paths with the maximum latency should be provided in the output file. The output file "sample.out" resulting from "sample.in" is given below.

sample.out	Comments
I1 G1 0 5 5 N	// Edge:<I1, G1>, Earliest time:0, Latest time:5, Slack:5, Critical:N
I2 G1 0 0 0 Y	// Edge:<I2, G1>, Earliest time:0, Latest time:0, Slack:0, Critical:Y
I2 G2 0 2 2 N	// Edge:<I2, G2>, Earliest time:0, Latest time:2, Slack:2, Critical:N
I2 G3 0 3 3 N	// Edge:<I2, G3>, Earliest time:0, Latest time:5, Slack:5, Critical:N
I3 G3 0 6 6 N	// Edge:<I3, G3>, Earliest time:0, Latest time:8, Slack:8, Critical:N
G1 G4 6 6 0 Y	// Edge:<G1, G4>, Earliest time:6, Latest time:6, Slack:0, Critical:Y
G2 G4 4 6 2 N	// Edge:<G2, G4>, Earliest time:4, Latest time:6, Slack:2, Critical:N
G3 G5 5 8 3 N	// Edge:<G3, G5>, Earliest time:5, Latest time:8, Slack:3, Critical:N
G4 G6 7 7 0 Y	// Edge:<G4, G6>, Earliest time:7, Latest time:7, Slack:0, Critical:Y
G4 G7 7 7 0 Y	// Edge:<G4, G7>, Earliest time:7, Latest time:7, Slack:0, Critical:Y
G5 G7 7 10 3 N	// Edge:<G5, G7>, Earliest time:7, Latest time:10, Slack:3, Critical:N
G6 G8 16 16 0 Y	// Edge:<G6, G8>, Earliest time:16, Latest time:16, Slack:0, Critical:Y
G7 G8 14 14 0 Y	// Edge:<G7, G8>, Earliest time:14, Latest time:14, Slack:0, Critical:Y
G8 O1 18 18 0 Y	// Edge:<G8, O1>, Earliest time:18, Latest time:18, Slack:0, Critical:Y
G7 O2 14 16 2 N	// Edge:<G7, O2>, Earliest time:14, Latest time:16, Slack:2, Critical:N
max latency: 19	// The maximum latency of critical paths
critical paths:	// List all critical paths
I2 G1 G4 G6 G8 O1	// I2→G1→G4→G6→G8→O1
I2 G1 G4 G7 G8 O1	// I2→G1→G4→G7→G8→O1

4 Command-line Parameter

In order to test your program, you are asked to add the following command-line parameters to your program.

[executable file name] [input file name] [output file name]

(e.g., StudentID.exe sample.in sample.out)

5 Submission Information

- Your program must be written in the C/C++ language and can be compiled on the Linux platform.
- The source files of your program must be named with "[your student ID].h" and "[your student ID].cpp".
- To submit your program, please archive all source files of your program into a single zip file, named "[your student ID].zip", and upload it to E3.

6 Due Date

Be sure to upload the zip file by "Wednesday, December 14, 2022". There will be a 25% penalty per day for late submissions.

7 Grading Policy

The programming assignment will be graded based on the following rules:

- Pass sample input with compilable source code (50%)
- Pass five hidden test cases (50%)

The submitted source codes, which are either copied from or copied by others, will NOT be graded.