Department of Electrical and Computer Engineering
University of Thessaly

ECE431 - CAD Algorithms

Fall Semester — Academic Year 2025-2026

Assignment 2^{nd}

Topological Graph Traversal & Unit Delay Annotation

28/10/2025 έως 7/11/2025

C. Sotiriou

Contents

1 Assignment Description

The goal of this assignment is to implement a topological traversal algorithm on a digital circuit represented as a directed acyclic graph (DAG). Students will annotate each gatepin with a level that indicates its logical depth and perform unit delay annotation to calculate timing delays throughout the circuit.

2 Background

In digital circuit design and analysis, understanding the timing and logical depth of signals is crucial for performance optimisation and correctness verification. Circuits are often represented as directed graphs, where nodes correspond to logic gates, flip-flops, or primary inputs/outputs, and edges denote the signal flow from one gate to another.

Fundamental in this context is determining the logical levels of each gatepin which reflect its position within the circuit hierarchy from the primary inputs. This information is essential when analysing the maximum propagation delay or performing timing verification, as it provides insight into how signals traverse the circuit and impact overall performance.

Topological sorting is a core technique used in processing DAGs, which represent combinational digital circuits. It ensures that each node (gate) is processed after all its dependencies (inputs) have been processed. This ordered traversal is crucial for assigning levels correctly, as it naturally respects signal dependency relationships.

Along with the topological sorting, level annotation assigns a value to each gatepin, indicating its distance from starting points, i.e. primary inputs and flip-flop output pins. Starting points start at level 0, and each subsequent gate's level is computed as one plus the maximum level among its input gates. This level annotation is necessary for identifying the critical / longest path of the circuit.

To analyse the timing behaviour accurately, unit delay modeling assumes each gate introduces a fixed delay (often one time unit for simplicity). By propagating these delays through the circuit, one can compute the maximum delay from inputs to outputs. The arrival time at each gatepin is calculated by adding the unit delay to the maximum arrival time of its input pins, enabling a clear estimation of the circuit latency.

This combination of topological traversal, level annotation, and delay propagation forms the foundation of static timing analysis. It provides a systematic framework for understanding and optimising circuit performance, especially in complex designs where timing constraints are critical.

3 Assignment Guidelines

- **Integration of Level and Unit Delay Annotation during Topological Traversal**

  The level annotation and unit delay calculations should be performed seamlessly as part of the topological graph traversal process. Specifically, when visiting each gate in the topological order, the algorithm should:

  1. Compute the level of the output gatepins as one plus the maximum level among its input gatepins.

2. Compute the gate's arrival time (unit delay) as one plus the maximum arrival time among its input gatepins.

Incorporating these annotations within the traversal ensures efficiency by avoiding multiple passes over the circuit graph.

- **Use of C++ Template Parameters for Delay Annotation**

  To promote modularity and make your code easily extended in the future (such as performing LUT-based delay annotation in a subsequent assignment), it is highly recommended to implement the delay annotation logic as a generic component.

  Specifically, the function responsible for topological traversal should be templated to access a delay annotation strategy as a template parameter. This could initially support:

  – Unit delay annotation, where every gate produces a fixed delay of 1 unit.

  – Future extension to LUT-based delay annotation, which will involve look-up tables for gate delay values.

  This approach encourages clean separation of traversal mechanisms from delay computation logic and fosters reusable, flexible code design.

- **Example on C17**

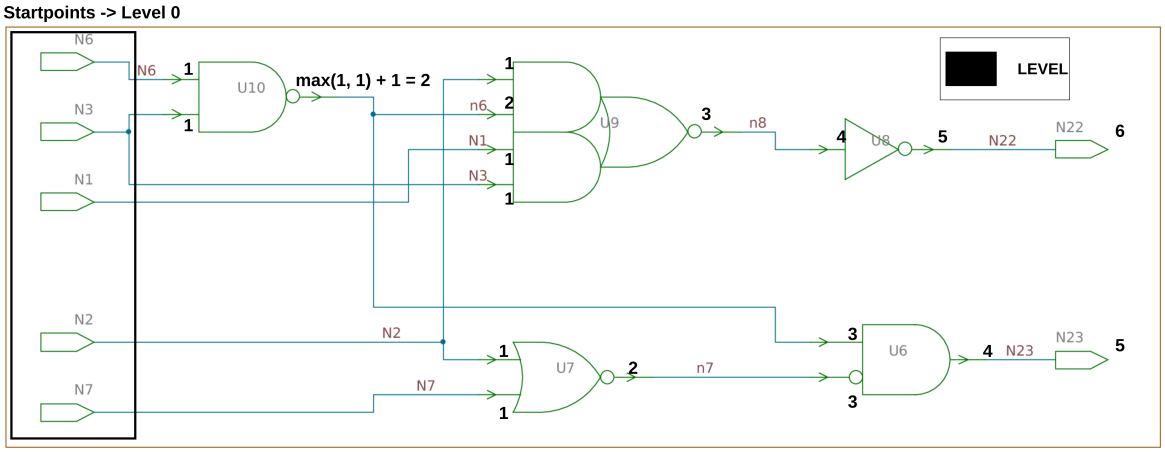

Figure 1: Level annotation on C17 circuit.

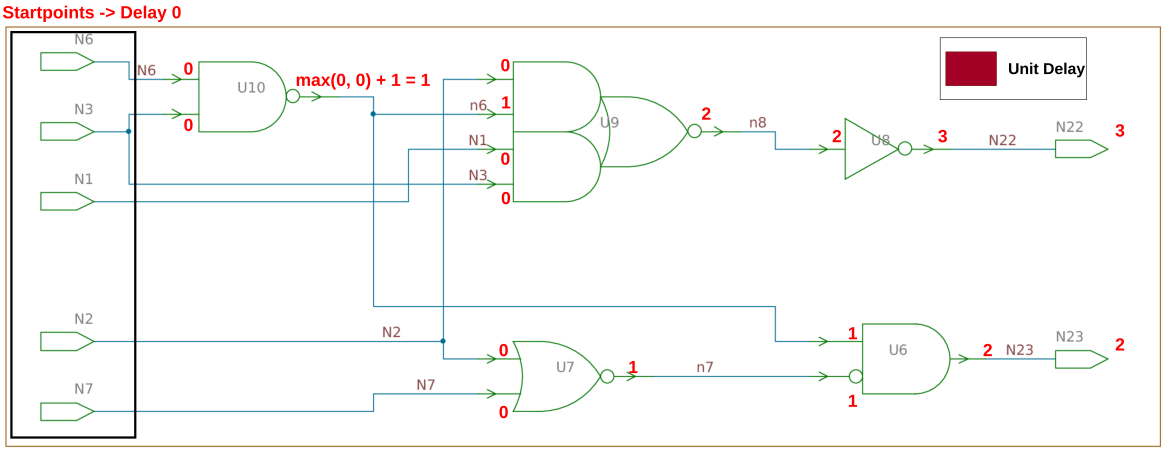

Figure 2: Unit Delay annotation on C17 circuit.

4 Assignment Objectives

1. **Topological Sort Traversal**

   Implement a topological sorting algorithm to order gates such that each gate is processed only after its input gates.

2. **Level Annotation**

   During the topological traversal of the circuit assign level to each circuit pins, where startpoints receive level 0, and each subsequent gatepins gets a level of one plus the maximum level among its preceding pins.
   Startpoints are the Primary Inputs of the top module and flip flop outputs.
   Endpoints are the Primary Outputs and Flip Flop inputs.
   Note: Check the LibCell class, which contains all information about Liberty cells. The field 'm_ff_group' is null for combinational cells.
   For sequential cells, it stores information about the clock and reset pins.

3. **Unit Delay Annotation**

   In addition to the level annotation, calculate the arrival time at each output pin by adding a unit delay of 1 to the maximum arrival time of its input pins. Note that for the purposes of this assignment the interconnect delay is not considered, and thus, the input pins of each gate have the same delay as the output pins of the driving gate. Finally the startpoints are assigned with an initial delay of 0.
   To calculate the delay on an output pin, we check all input pins that have a timing arc to this output and take the maximum delay among them, then add 1. **Use function "getFromToTimingArcs" to check if there is timing arc from input to output pin**.

   Example:

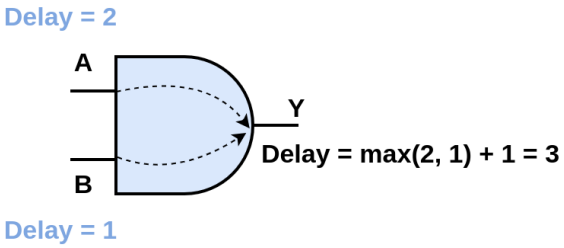

Figure 3: Cell with all timing arcs

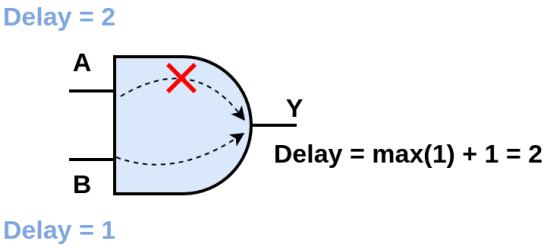

Figure 4: Cell with one timing arc

4. **Output format**

   You need to create a TCL command **"calculate_pins_delay_and_level"** (check file CADI_project.cpp how it creates the TCL functions) which will call your implementations for toposort, level annotation and delay annotation. Also, this function should print all pins sorted by name in the exact following format:
   "fullpinname <level> <delay>"
   Note: Module ports (except those of the top module) should be skipped during level and delay annotation. Their level and delay values must be set to -1, and they should not be printed in the output.

5 Deliverables

Students are required to submit the following for sucessful completion of this assignment:

1. **Code Changes**

   - Submit only the specific code changes made for this assignment using git (we will upload more info about this).
   - The changes should include the implementation of the topological traversal, level annotation, and unit delay annotate functionality.
   - Additional changes containing implementations for previous assignment should be included for evaluation.
   - Ensure that commits are well-documented with meaningful comments.
   - Include an automated script that loads all designs into the tool and, for each design, exports a separate file containing the level and delay information as previously described.

2. **Presentation**

   - Prepare and deliver a presentation showcasing your implementation.
   - The presentation should include:
     - Explanation of the topological traversal algorithm used.
     - Details on how level and unit delay annotations are integrated during the traversal.
     - Discussion of design decisions and challenges faced.
     - Live demo running the tool for some of the provided designs.
     - Profiling data of your code about runtime and memory

6 Evaluation

- **Correctness of Annotations**
  - The primary metric for evaluation is the accuracy of level and delay annotations for each gatepin in the provided test circuits.
  - A bonus will be awarded to those who attempt to optimise their data structures and algorithms for runtime and memory efficiency.
- **Code Quality and Structure**
  - Code must be modular, clean, and well-documented, demonstrating proper integration of level and delay annotation logic during topological traversal.
  - Efficient and effective use of C++ templates for delay annotation will be assessed based on design clarity and future extensibility.
- **Presentation and Communication**
  - The presentation you submit on eClass will also contribute to the overall evaluation of your work.

7 Acknowledgment

We would like to thank Synopsys® for providing us the software as well as the liberty parser.