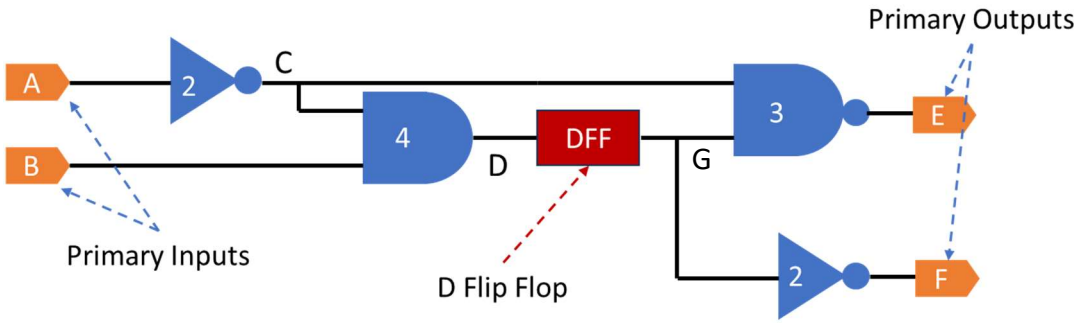
COL215: Digital Logic and System Design
Software Assignment 2
Submission Deadline: 15th October 2023, 11:55 PM

In this assignment, we will generalise our earlier delay analysis program of Assignment 1 in two different directions: (1) Circuits with D Flip Flops and (2) optimising for area.

Part 1: Circuits with D Flip Flops

Consider the following variation of the circuits we handled in Assignment 1, with D Flip Flops (DFF in the figure below). The D Flip Flop has one input and one output. We will discuss the DFF internals in detail later in the course, but for now, let us assume that it has zero propagation delay. However, it has the property that the signal arriving at its input represents the END of a combinational path, and the signal at its output represents the START of a new combinational path. Note that the DFF output should be considered available at time 0 as it signifies the beginning of a new combinational path. In this scenario, what is the delay of the longest combinational path in a given circuit? We need to examine the following paths:

- From any primary input to any primary output (not passing through a DFF)
- From any primary input to any DFF input
- From any DFF output to any primary output
- From any DFF output to any DFF input



Modify your earlier program to report the longest combinational delay in a circuit with DFFs.

In the figure above, the longest combinational path delay is 2+4 (from A to the DFF input). Note that a DFF separates the AND and NAND gates above, so they are never part of the same path. Note also that our circuits are now allowed to contain cycles, as long as the cycle contains at least one DFF – the cycle essentially contains combinational paths starting at the output of a DFF and terminating at the input of a DFF.

Input: Circuit file (see example file *circuit.txt* for the diagram above)
Input: Gate Delay file (see example file *gate_delays.txt* for diagram above)
Output: Longest Combinational Delay file (see example file *longest_delay.txt* for diagram above)

Part 2: Area Optimisation

Suppose we have a choice of gate implementations. Each gate (e.g., AND2 gate) now has 3 implementation choices: fast/slow/moderate. Corresponding to these, the gates have 3 areas: large/medium/small. The fastest gate implementation also has the largest area. Design and implement the solution to the following problem: find the smallest circuit that respects a given longest combinational path delay constraint. That is, select the implementation for each gate in the circuit, such that the total area (= sum of gate areas) is minimum and the longest combinational path delay is less than or equal to a given constraint.

Input: Circuit file (see example file *circuit.txt* for the diagram above)
Input: Gate Delay file (see example file *gate_delays.txt* for diagram above)
Input: Delay Constraint file (see example file *delay_constraint.txt* for diagram above)
Output: Minimum Area file (see example file *minimum_area.txt* for diagram above)

File Formats and Example

- The format of the ***gate_delays.txt*** file has now changed from:

  NOR2 3.5
  format: <type> <delay>

  To:

  NOR2_1 NOR2 3.5 10
  NOR2_2 NOR2 3 12.5
  NOR2_3 NOR2 2.5 15
  ...etc.
  format: <implementation> <type> <delay> <area>

  You can use the FASTEST gate implementation for answering Part 1 of the assignment.

- The format of ***circuit.txt*** remains the same as Software Assignment 1 except that now, there could be an additional line of the form

  DFF <input signal> <output signal>

- The ***delay_constraint.txt*** contains a number indicating the longest combinational path delay constraint.
- The ***longest_delay.txt*** should contain a single number, the answer of Part 1.
- The ***minimum_area.txt*** should contain a single number, the answer of Part 2.

Example for Part 2:

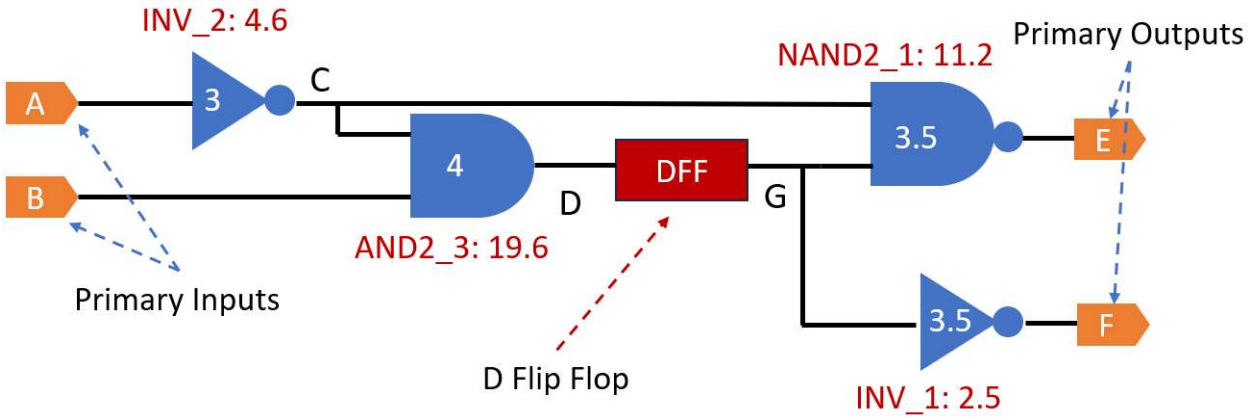
Consider the following gate implementations (provided in *gate_delays.txt*) for the above circuit:

NAND2_1 NAND2 3.5 11.2
NAND2_2 NAND2 3 13
NAND2_3 NAND2 4.5 5.3
AND2_1 AND2 16.2 9.5

AND2_2 AND2 7 12
AND2_3 AND2 4 19.6
NOR2_1 NOR2 3.5 10
NOR2_2 NOR2 3 12.5
NOR2_3 NOR2 2.5 15
OR2_1 OR2 4.5 12.8
OR2_2 OR2 7.5 10.3
OR2_3 OR2 3.75 17
INV_1 INV 2 7.33
INV_2 INV 3 4.6
INV_3 INV 3.5 2.5

and delay constraint = 7 (provided in *delay_constraint.txt*)

Then, the optimal choice of gate implementations to minimise total area would be:



And the **minimum area** = **4.6 + 19.6 + 11.2 + 2.5 = 37.9** is the answer to Part B for the given inputs.

Use Python or C/C++ for implementation. Submit the following:

(a) A short document with the following details:
   a. Your algorithm.
   b. Explain the time complexity of your algorithm. More efficient algorithms and faster implementations will receive higher marks, so optimize your implementations.
   c. Your *testing strategy*. What test cases did you write to check your implementation? Why? Make sure that the test cases validate the features of your algorithm.

(b) Your implementation.
   a. Code
   b. Test cases