Software Assignment - 2 - Report

COL215 - Digital Logic and System Design

Nikhil Zawar - 2022CS11106
Amaan Ali Sayed - 2022CS11618

1 PART 1 - Circuits with D Flip Flops

1.1 Problem Statement

The problem statement is a continuation to the assignment 1. Here we have to calculate the longest combinational path delay in a circuit with gates and flip flops. DFFs (D Flip Flops) are considered to have zero propagation delay. They have 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. The DFF is considered available at time 0 as it signifies the beginning of a new combinational path.

1.2 Implementation

The implementation is done in Python. We have implemented a graph data structure where in the inputs, outputs and signals are vertices; and the gates and DFFs are edges. The Edges are weighted and the weight of the edge represents the delay that is caused when a signal passes through that gate(edge). Since according to our input file each gate has three kinds of delay. For the sake of first part we will consider the minimum of all the three delays for a particular gate. The key element of our algorithm is the Depth First Search Traversal in order to find the longest combinational delay. When two signals are input in a gate, then consider the maximum delay that can happen in producing the output. This is recursively implemented with the termination of arriving at the primary outputs. This will result in the correct delay when we finally reach a primary output.

1.3 Time Complexity Analysis

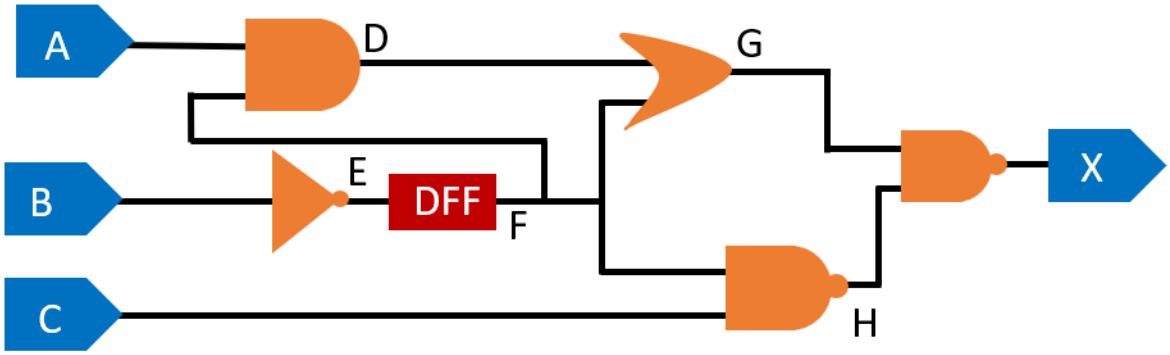
There are two aspects of time complexity. One is the use of python functions to read the input files, parse the input files, build a graph and print the output file. The other important aspect is the time complexity involved in the calculation of the longest combinational path delay. The first part takes a time complexity of O(n), where n is the number of lines in input files, or the lines to be parsed individually for both the tasks. The main time complexity analysis required is for the algorithm involving the calculation of the longest combinational path delay. We have used the Depth-First Search Algorithm that traverses in the graph as far as possible along each branch before backtracking. While traversing through the graph we visit the vertices and the edges both, so we tell the time complexity is O(V+E), where

V is the number of vertices and E is the number of edges. It is important to note that we might visit a particular vertex or edge more than once, and this is taken into account since the time complexity is a asymptotic analysis of the algorithm.

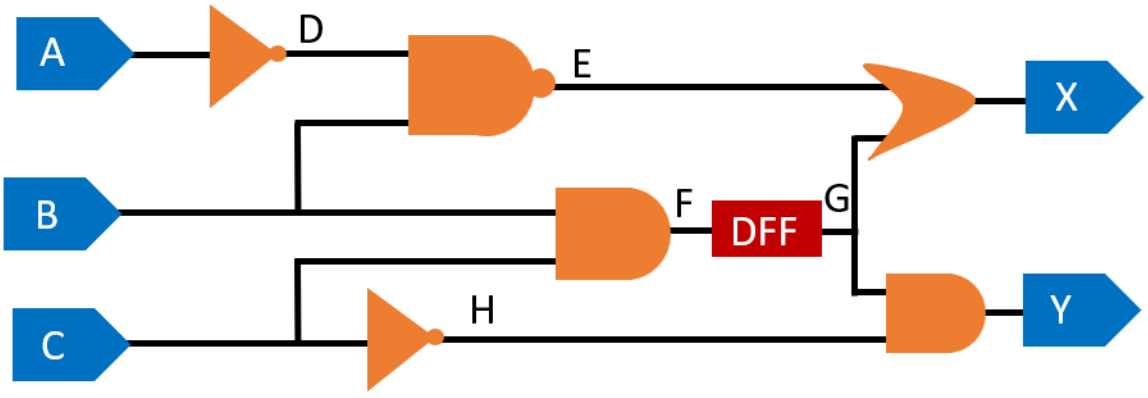
1.4 Testing Strategy

In addition to the given test cases, we have developed the following test cases to check our implementation. The test cases do validate the algorithm. The test cases also consist circuits that are allowed to have cycles. The cycles necessarily contains at least one DFF which means that the cycle is moreover a combinational path which starts at one end of a DFF and ends at the other end.

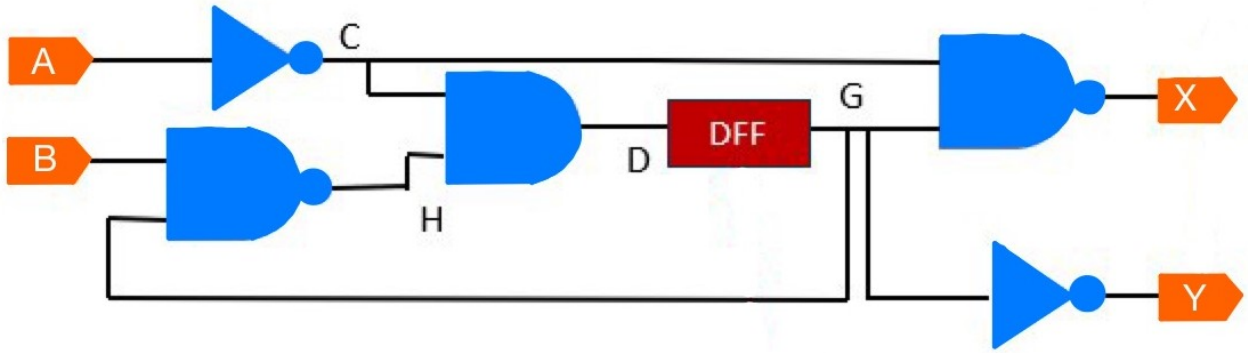
1.4.1 Test Case 1



1.4.2 Test Case 2



1.4.3 Test Case 3



2 PART 2 - Area Optimisation

2.1 Problem Statement

Here we have two goals to achieve. One is to find the circuit with longest combinational path delay that is less than or equal to a given constant, which we read from a new, third input file - Delay constraint file. The other goal is to minimise the total gate area. We basically have 3 implementations of a particular gate - fast/moderate/slow and corresponding to these the gates have 3 areas large/medium/small respectively. The fastest gate implementation also has the largest area. The problem is to find the circuit whose sum of all gate areas is minimum and it also follows the given longest combinational path delay constraint.

2.2 Implementation

Here again we have used a very similar implementation to that in part 1. We have implemented a graph data structure which stores the circuit connections. The primary inputs, outputs and signals are visualized as the vertices in the graph. The gates and DFFs are visualized as the edges of the graph. It is a weighted graph, but here the edges have two attributes as weights. These two attributes are the gate delays and the area occupied by the gates. Our implementation is partially brute force. Make note of each possible permutation and then check the conditions. All of those permutations with longest combinational delay less than or equal to delay constraint are useful. Among all such permutations we need the one with the minimum area. The method used to find all the permutations is a recursive function which calls the permutations while recursively building it up.

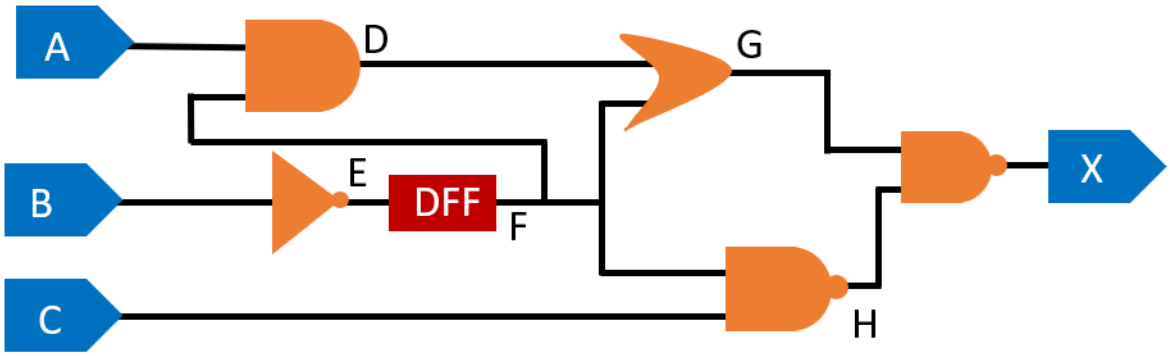
2.3 Time Complexity Analysis

The time complexity analysis to find the combinational delay in a particular permutation of the gates remains the same as for part 1. Here the addition is the calculation of areas in the same path. Since we are doing this through a depth first traversal the asymptotic time complexity remains O(V+E), where V is the number of vertices and E is the number of Edges in the graph. But after this step, we need the circuit satisfying the given constraint and minimum area. To do this, we have considered each possible permutation in our implementations. Let's say n is the number of gates in the circuit. Each gate has three variations - large/moderate/small. So we have three possible permutations from a single gate keeping the configuration of the other gates fixed. Similarly for every gate when we extend the same argument we can say that the complexity will be of the order of $n*3^n$. So the algorithm takes O($n*3^n$) time to evaluate the final output.

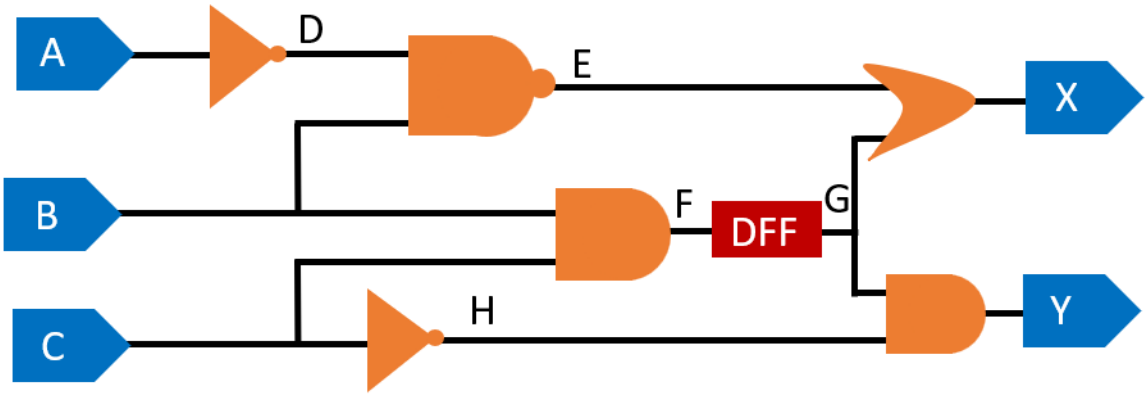
2.4 Testing Strategy

In addition to the given testcases, we have developed the following test cases to check our implementation. The test cases do validate the algorithm. The test cases also consist circuits that are allowed to have cycles. The cycles necessarily contains at least one DFF which means that the cycle is moreover a combinational path which starts at one end of a DFF and ends at the other end.

2.4.1 Test Case 1



2.4.2 Test Case 2



2.4.3 Test Case 3