Report

COL215 Software Assignment 1

Jaskaran Singh Bhalla (2021TT11139) Saurabh Arge (2022CS11610)

Algorithm

For Part A

We have used a recursion based approach to calculate the required output delays. We make a function call for each primary output which based upon the type of the gate makes the same function call for its input node and then returns the maximum of the delays of its input nodes. Recursively, each node makes a call for its input node unless we reach the primary input nodes which have a zero delay and we return 0. At each step the recursion call returns the max of the delays of the input nodes plus the delay of the node itself. So we have solved the problem in the backward direction of the circuit.

For Part B

We have used a similar recursion based approach in this as well to calculate the required input delays. We make a function call for each primary input which based upon the type of the gate makes the same function call for its output reference nodes. This continues until we reach the output node and for each output node we know the expected delay and we return that. At each point where the recursion we subtract the delay of the current node and then return the minimum of all the output paths. So we have solved the problem in the forward direction of the circuit by making recursion calls.

Time Complexity

For Part A

Since we have used a recursion based approach, time complexity will largely vary based upon the design of the circuit. The worst case time complexity will arise when the circuit recursion tree is highly unbalanced leading to higher depth of recursion and multiple recursive calls per gate. Since for each node there can be let us say M inputs and during recursion we visit each node once. So time complexity for the code will be of the order of O(M x N). But if we assume that each node can have at max only two input nodes i.e. M = 2 then the time complexity of our code will be O(2 x N) which is O(N). So we have arrived at a linear time complexity for the program.

For Part B

In this case also, we have used a similar recursion based approach, time complexity will largely vary based upon the design of the circuit. The worst case time complexity will arise when the circuit recursion tree is highly unbalanced leading to higher depth of recursion and multiple recursive calls per gate. Since for each node there can be let us say M output references and during recursion we visit each node once. So each node can make at max M sub recursion calls. So time complexity for the code will be of the order of O(M x N). But if we assume that each node can have at max two output references i.e. M = 2 then the time complexity of our code will be O(2 x N) which is O(N). So we have arrived at a linear time complexity for the program.

Note : The analysis given at last is just an assumption. The original time complexity for our code is O (M x N). Also this time complexity can be made better by reducing the number of computations by using memoization which we haven't been introduced to in the course, so we have refrained ourselves from implementing it.

Finally, time complexity for both the parts is O(M x N) where M is the maximum number of input nodes or output references and N is the number of nodes in the circuit.

Testing Strategy

We tested the code on the given sample test case. We have created test cases where we have given input to multiple gates and have multiple outputs. We have considered the cases where we have consecutive gates, two gates taking the same input, a single output going to multiple other gates to multiple outputs. A test case with all zero delays, a single input given and floating point delays. The test case where one output is received. As we have used a recursion based approach, so we have used test cases with deep recursion levels as well. We used test cases with randomised delays and gates as well.