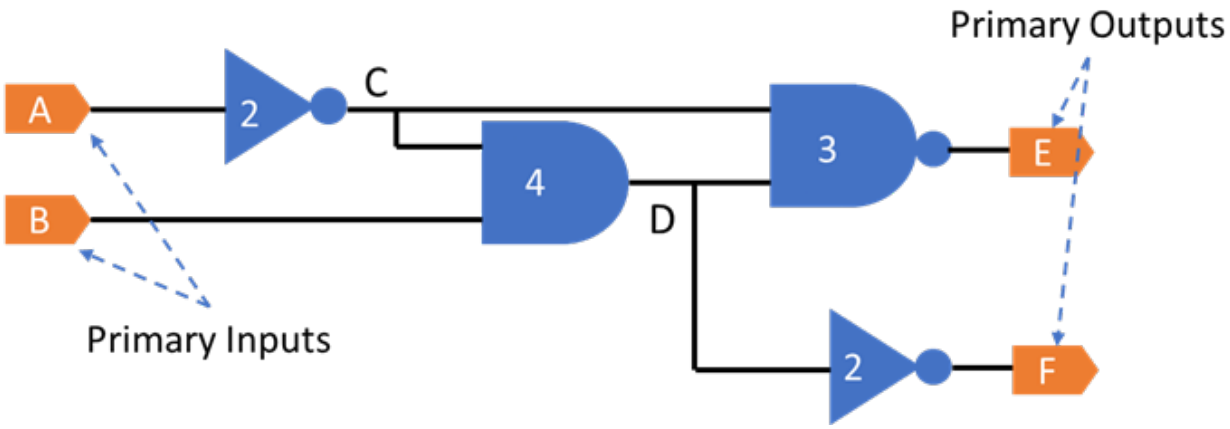
COL215 SW Assignment 1

Input and Output Delays

Jahnabi Roy : 2022CS11094
Anamika : 2022CS11098

15th August, 2023



1 AIM

The project is divided into the following two parts :

1.1 Calculating Output Delays of a circuit:

(a) To calculate how much time a circuit takes to compute its primary outputs? Write a program to compute the delay of every primary output in a combinational circuit (defined as the earliest time when the output is ready), given the circuit representation and delay information of the individual gates.

Input: Circuit file (see example file *circuit.txt* for diagram above)

Input: Gate Delay file (see example file *gate_delays.txt* for diagram above)

Output: Output Delay file (see example file *output_delays.txt* for diagram above)

1.2 Calculating Input Delays, given the output delays of a circuit:

(b) Extend program to answer the converse question: suppose we require the output to be ready at a specific time. When do we require each input to be ready so that the output timing requirement is met?

Input: Circuit file (see example file *circuit.txt* for diagram above)

Input: Required Output Delay file (see example file *required_delays.txt* for diagram above)

Output: Input Delay file (see example file *input_delays.txt* for the diagram and *required_delays.txt* file above)

2 OUR ALGORITHM

2.0.1 READING THE INPUT FILE AND STORING THE DATA

We first start by reading the *circuit.txt* file. We create a dictionary named *store_def* to separately store the Primary inputs, Internal signals, and Primary outputs. We then create another dictionary *input_output* and initialize the starting time of all the signals to 0. Then we start reading the *gate_delays.txt* file. We create a dictionary *gate_delay* to store the value of time delays for each gate.

2.0.2 SORTING THE CIRCUIT

We read the file *circuit.txt* and store the gate-signal relations in the list named *unsorted_circuit*. Now we loop over the unsorted circuit and sort it according to the basic logic that a gate cannot operate if the inputs have not been defined before in the circuit; that is if any of the inputs is neither one of the primary inputs nor the result of a previous operation. For this, we maintain a set named *existing_signals*. The only existing signals would be the primary inputs at the very start. And on gate operation, the outputs generated would then be added to the set of existing signals.

For updating this set, named *existing_signals*, we keep a variable called flag and run a loop over the *unsorted_circuit* list. If in *unsorted_circuit* that is also a list of lists (gate-signal relations), we check the inputs of each such relation. If any of the inputs do not exist in the *existing_signals* set, we flag the relation one and check for the following gate-signal relation. However, for the case where all the inputs exist in the existing signals, we append it into a new list called *sorted_circuit* and remove it from the unsorted circuit. We also update the length of the *unsorted_circuit* and store it in variable g, and then we start looping over the

unsorted_circuit list from index 0.

2.0.3 LOOP INVARIANT

For the j^{th} step of the iteration of *unsorted_circuit* of length g, we append the gate-signal relation iff it occupies the same place in the circuit where all its inputs have been defined in the *existing_signals* set.

If the circuit is valid, then ultimately all the gates have a place in the circuit. On the other hand, if it is not valid, then the length of *unsorted_circuit* stored in variable g, never becomes zero and we simply exit from the program.

2.1 PART A

2.1.1 COMPUTING THE DELAY OF THE OUTPUTS

If the gate was an inverter, we simply added the gate delay value to the time at which the input was available and added it to the time at which the output is available when existing output value is zero. Else we take the minimum of the existing value and sum of gate delay and input.

For the case of other gates, we are given two inputs. So, we take the time at which all the inputs are available, that is maximum of the times at which inputs are available and added the *gate_delay* to the time at which the output is available.

2.1.2 WRITING THE OUTPUT

We create a file named *output_delays.txt* and write the obtained output delays.

2.2 PART B

2.2.1 UPDATING THE GIVEN OUTPUT DELAY VALUES

We now read the *required_delays.txt* and update the time stamps of the primary outputs to the time at which they are required to be available. Then, we reverse the sorted circuit, so that we can traverse it backwards to get the required input delays. If the gate was is an inverter, we take the output of the relation and subtract the gate delay and store it in the time at which the inputs are available. For other gates, if time stamp of the inputs are not 0, then we take the minimum of this time stamp and the result of subtraction of *gate_delay* from the output. And if it is equal to 0, then we simply update it it as the result of subtraction of *gate_delay* from the output.

If the input delays of the primary inputs are negative, then no solution exists. Else, it is possible to get the outputs at the provided time.

2.2.2 WRITING THE OUTPUT

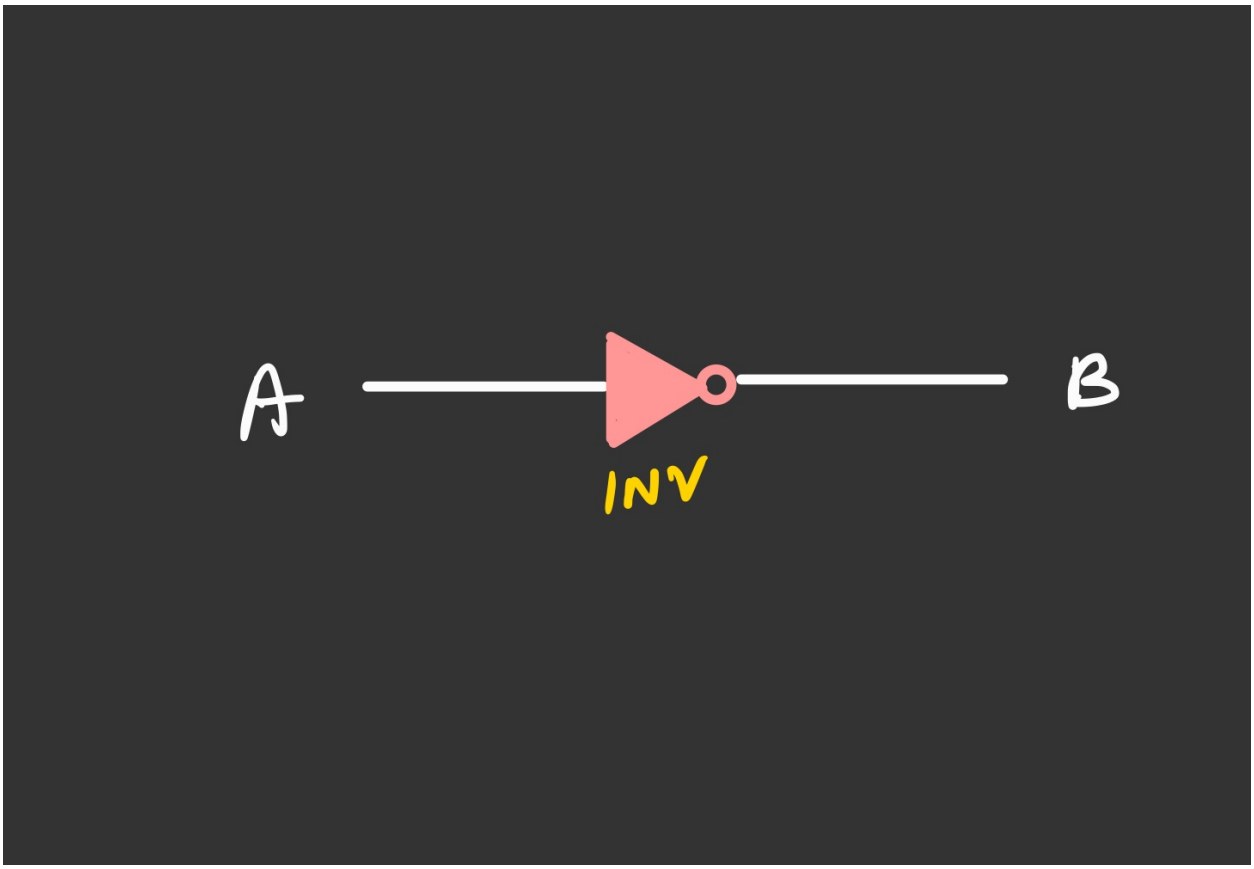
We create a file named *input_delays.txt* and write the desired output.

3 TEST CASES

3.1 Test Cases 1.0

3.1.1 CHECKING FOR A CIRCUIT WITH NO INTERNAL SIGNALS

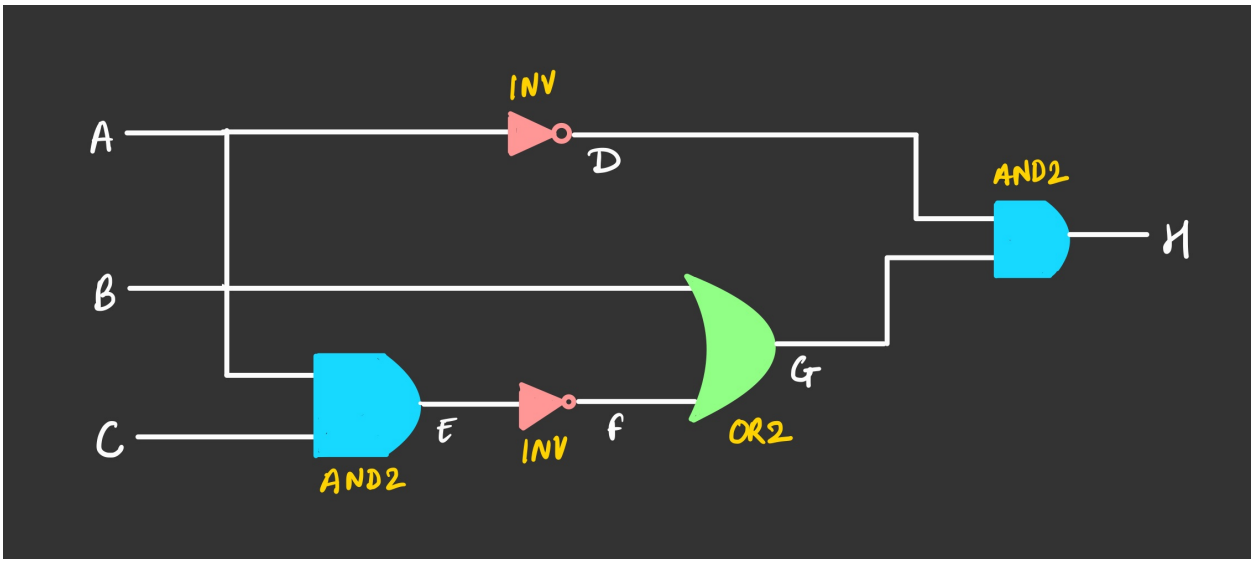
We have a simple circuit containing only one inverter gate with one primary input and one primary output gate. Our code successfully worked on such an circuit.



3.2 Test Cases 2.0

3.2.1 CHECKING FOR A CIRCUIT WITH GATES WHOSE INPUTS AND THE OUTPUTS ARE INTERNAL SIGNALS

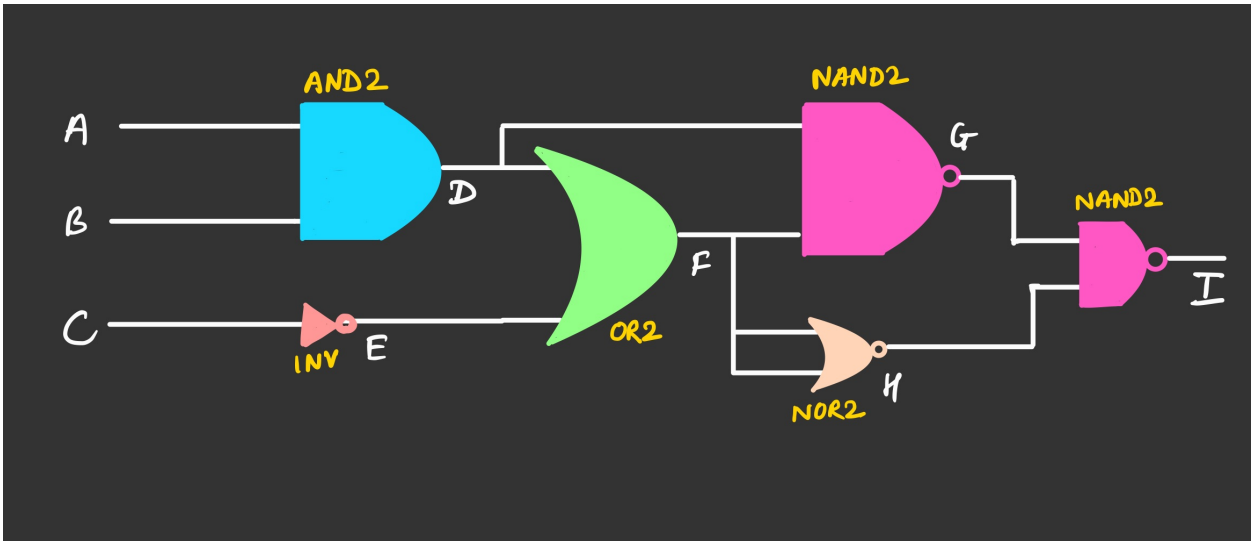
We have a circuit containing an inverter gate with whose input and output are internal signal. Our code successfully worked on such an circuit as well.



3.3 Test Cases 3.0

3.3.1 CHECKING FOR A CIRCUIT WITH GATES WHOSE BOTH THE INPUT SIGNALS ARE SAME

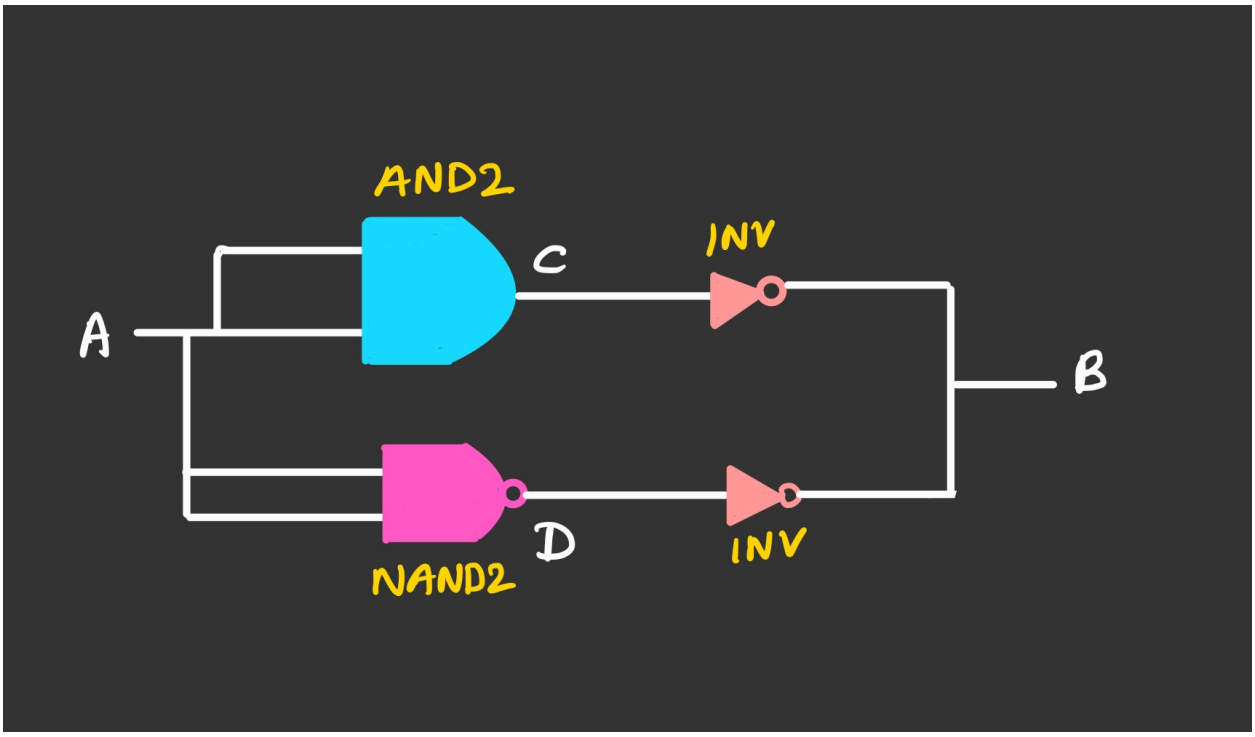
We have a circuit containing an AND gate whose both the input signals are same. Our code successfully worked on such an circuit as well.



3.4 Test Cases 4.0

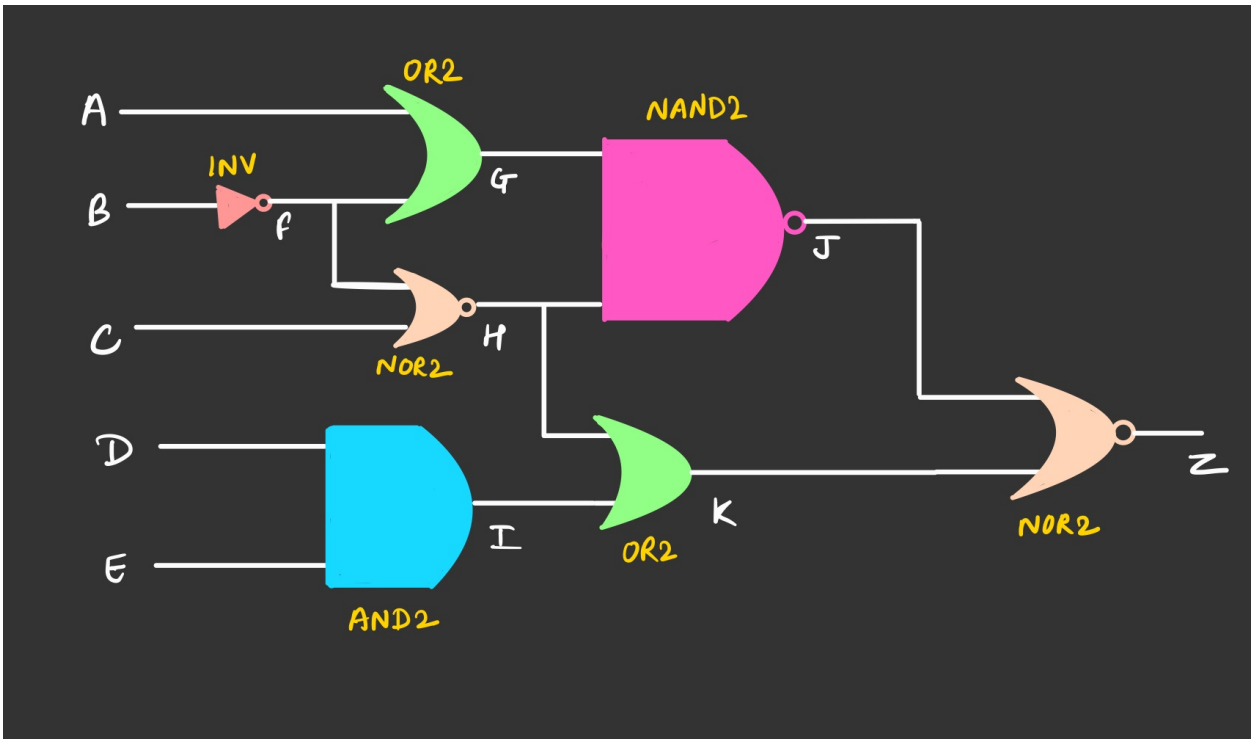
3.4.1 CHECKING FOR A CIRCUIT IN WHICH THE OUTPUTS BY DIFFERENT PATHS REACH A NODE

We have a circuit in which there are several gates whose outputs reach the same node. In this case the timestamps at which these outputs reach the node may be different so we took the time at which all the output from different paths have reached. Our code worked well for this circuit as well.



3.5 Test Case 5.0

Checking for all gates and a more complex circuit.



4 TIME COMPLEXITY

4.1 Time complexity of arranging the gates in the correct order:

4.1.1 WORST CASE

In our algorithm as we mentioned that we are looping over the unsorted_circuit list to get the position of each gate relation in the sorted_circuit list. In this we start checking the unsorted_circuit list from index 0 and if we encounter a gate with all the inputs in the existing_signal list then we append it to the sorted_circuit list and remove it from the unsorted_circuit list and start looping the updated unsorted circuit_list from index 0 again. So, if we consider the worst case in which the gates are given in the reverse order in which they occur in the circuit and the no of signals in the circuit are of the3 order of inputs then

we will have to iterate over a list of length n then the length of unsorted_circuit will become $n-1$ and it will go on till the length of unsorted_circuit becomes zero.

In total we will have to iterate

n+(n-1)+(n-2)+(n-3)+.....+2+1 = $\frac{n(n+1)}{2}$ times.

And in each of iteration we will have to check whether the signal exist in existing signals or not. Hence, worst time complexity of algorithm will be $O(n^3)$

4.1.2 BEST CASE

The best time complexity of our algorithm would be O(n) and it would be in the case when the unsorted_circuit list is having the gates in the correct order and the number of signals is not of the order of the input(no of gate-signal relations) as in that we will have to iterate over the unsorted_circuit just once.

4.2 Time complexity of calculating output delays:

To calculate output delays we iterate the sorted_circuit once while calculating the time at which the signals are available. Hence, this is of time complexity of O(n).

4.3 Time complexity of Part A:

So, the average time complexity of Part A is O(n^2)+O(n)=O(n^2)

Worst case time complexity of Part A is O(n^3)+O(n)=O(n^3)

Best case time complexity of Part A is O(n)+O(n)=O(n)

4.4 Time complexity of calculating input delays:

To calculate input delays we iterate the reversed sorted_circuit once while calculating the time at which the inputs of each gate are available. Hence, this is of time complexity of O(n).

4.5 Time complexity of Part B:

So, the average time complexity of Part B is O(n^2)+O(n)=O(n^2)

Worst case time complexity of Part B is O(n^3)+O(n)=O(n^3)

Best case time complexity of Part B is O(n)+O(n)=O(n)

5 Testing Strategy

To check our code we checked with it different sort of the circuits which are possible without the cycles. We included test cases with only single gate i.e. no internal signals, with circuit of higher level that is there are a sequence of gates in between primary input receiving gates and primary output returning gates i.e. have gates whose inputs as well as outputs are internal signals. We even checked if our code worked if the both the inputs of a gate are same. We checked our code on the circuit in which output from several gates reach a node at different times. In this way our test cases cover all the different types of connection possible in a circuit without cycles.