RISHIK SAHA-2022CS11932| PARTH VERMA-2022CS11936

COL215-SOFTWARE_1

REPORT

PART-B

ALGORITHM:

The content from the given input files is extracted and stored in dictionary (for comfortable retrieval) named "data".

Basically, here we create a forward directed graph representing the circuit and use recursion to get the result. Now when we want the delay of a node say 'n':

We call a function calc_delayB(n) which does the following:
If the node is a primary output:

we return the required output from "required_outputs" file.

Else:

We access all the connected nodes to n (through the dictionary "data" in O (1)) We the find the minimum delayB among them by recursively calling calc_delayB() for each of them. Then the corresponding gate delay is

subtracted from the minimum value obtained and the result is returned.

Finally we call the calc_delayB() function for the input nodes and get the desired input_delays.

TIME COMPLEXITY:

The time complexity of the above algorithm will be

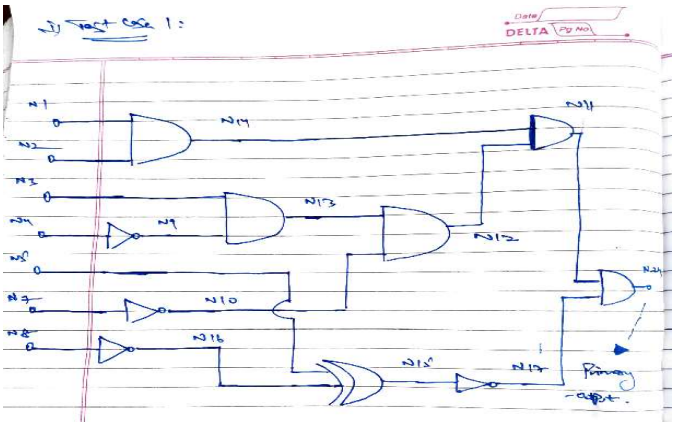
O(v+e) where v is the no of vertices and e is the no of edges in the graph.

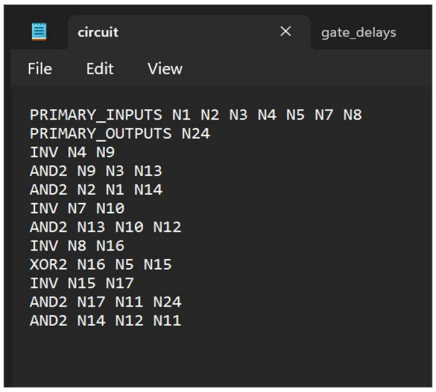
This is because for each node we recursively call the calc_delayB () function and hence as a result travel all the possible paths of the graph.

TESTING STRATEGY:

Test Cases:

Test Case 1:





input _delays

File Edit View

```
N1 7.0
N2 7.0
N3 3.0
N4 1.0
N5 10.5
N7 5.0
N8 8.5
```

Expected output:

output_delays

File Edit View

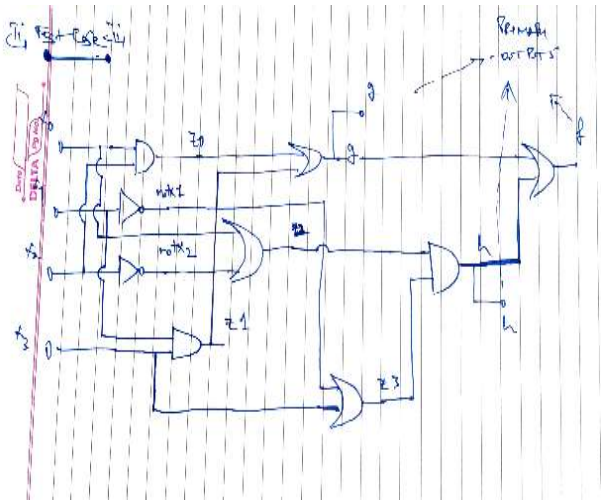
```
N24 18.0
```

gate_delays

File Edit View

```
NAND2 3
AND2 4
NOR2 3.5
OR2 4.5
INV 2
XOR2 2.5
```

Test Case2:



circuit

File Edit View

```
PRIMARY_INPUTS x0 x1 x2 x3
PRIMARY_OUTPUTS g f h
INV x2 notx2
INV x1 notx1
AND2 x0 x2 z0
AND2 x1 x3 z1
OR2 x0 notx2 z2
OR2 notx1 x3 z3
OR2 z0 z1 g
AND2 z2 z3 h
OR2 g h f
```

gate_delays (1)

File Edit View

```
NAND2 3
AND2 4
NOR2 3.5
OR2 4.5
INV 2
XOR2 2.5
```

<u>Expected output:</u>

output_delays

File Edit View

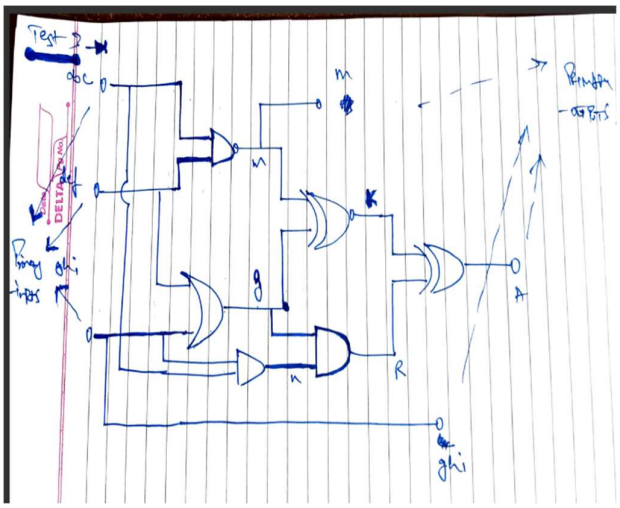
```
g 8.5
f 15.0
h 10.5
```

input_delays

File Edit View

```
x0 8.5
x1 6.5
x2 6.5
x3 8.5
```

Test Case _3:



circuit

File Edit View

```
PRIMARY_INPUTS abc def ghi
INTERNAL_SIGNALS g n R k
PRIMARY_OUTPUTS m A ghi
NAND2 abc def m
OR2 def ghi g
AND2 ghi abc n
AND2 g n R
XNOR2 m g k
XOR2 k R A
```

gate_delays

File Edit View

```
NAND2 3
AND2 4
NOR2 3.5
OR2 4.5
INV 2
XOR2 2.5
XNOR2 3
```

<u>Expected output:</u>

output_delays

File Edit View

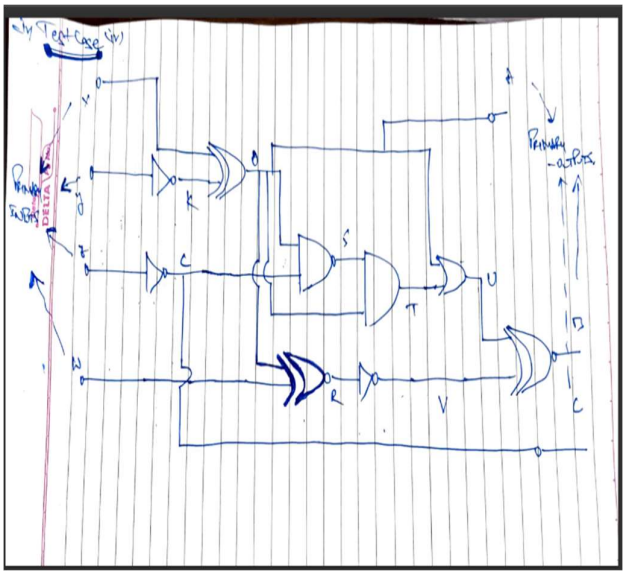
```
m 3.0
A 11.0
ghi 0
```

input_delays

File Edit View

```
abc 0.0
def 0.0
ghi 0.0
```

Test Case _4:



circuit

File Edit View

```
PRIMARY_INPUTS x y z w
INTERNAL_SIGNALS A k C S T R V U
PRIMARY_OUTPUTS A B C
INV y k
XOR2 x k A
INV z C
XNOR2 A w R
NAND2 A C S
AND2 A S T
INV R V
OR2 A T U
XNOR2 U V B
```

gate_delays

File Edit View

```
NAND2 3
AND2 4
NOR2 3.5
OR2 4.5
INV 2
XOR2 2.5
XNOR2 3
```

<u>Expected output:</u>

gate_delays

File Edit View

```
A 4.5
B 19.0
C 2.0
```

input_delays

File Edit View

```
x 2.0
y 0.0
z 0.0
w 2.0
```

Reason for choosing the above Test Cases:

The above testcases were chosen for testing the algorithm in different scenarios.

The test cases ensured that our algorithm manages sufficiently high no of nodes and complex graphs.

The nodes were given variable names (multi-character) to ensure there is no error in reading of input files.

All possible gates including (XOR2, XNOR2) were introduced in the test cases.

The test cases were designed so that a node is connected to multiple nodes (in both forward and backward directed graphs) which was essential in testing the recursive algorithm.

The output signals were taken from all the regions of the circuit (input stage, middle region, ending region) of the graph.