RISHIK SAHA-2022CS11932| PARTH VERMA-2022CS11936

COL215-SOFTWARE_2

REPORT

PART-A

ALGORITHM:

The content from the given input files is extracted and stored in various maps for comfortable retrieval.

Basically, here we create a backward directed graph representing the circuit and use recursion to get the result. Now when we want the delay of a signal say ‘n’:

We call a function calc_delayA(n) which does the following:
If the signal is a primary input:

we return 0. (A zero delay)

if the signal is an input from D Flip Flop

we return 0.

If not:

We retrieve the connected nodes to ‘n’ through map named “connections_A” in O (1) and find the maximum delay among them through recursively calling the calc_delayA () function for each of them. Further we

add the delay of the corresponding gate and return the sum.

Further we maintain a vector named “results” where we store the delays of all possible combinatorial paths by calling calc_delayA () for circuit outputs (primary outputs and the signal going into a D Flip Flop)

We then extract the maximum delay and print it in

“longest_delay.txt”.

TIME COMPLEXITY:

The time complexity of the above algorithm will be

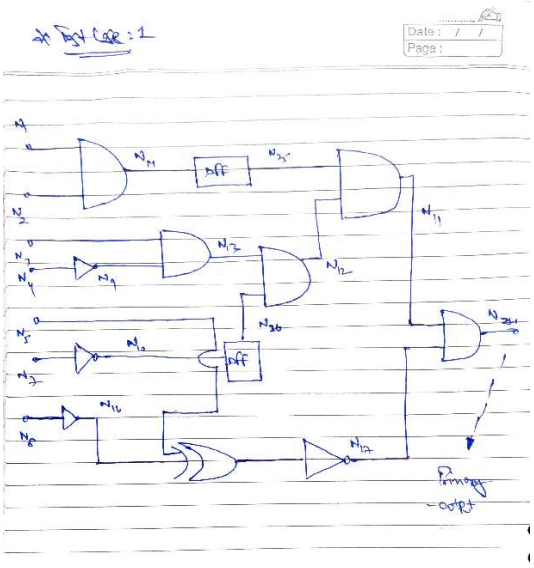
O(v+e) where v is the no of vertices and e is the no of edges in the graph.

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

TESTING STRATEGY:

Test Cases:

Test Case 1:



circuit

File Edit View

```
PRIMARY_INPUTS N1 N2 N3 N4 N5 N7 N8
PRIMARY_OUTPUTS N24
INTERNAL_SIGNALS N6 N9 N10 N11 N12 N13 N14 N15 N16 N17 N18 N19 N20 N21 N22 N23 N25 N26
INV N4 N9
AND2 N9 N3 N13
AND2 N2 N1 N14
INV N7 N10
AND2 N13 N26 N12
INV N8 N16
XOR2 N16 N5 N15
DFF N14 N25
INV N15 N17
DFF N10 N26
AND2 N17 N11 N24
AND2 N25 N12 N11
```

gate_delays

File Edit View

```
// Format:
// GATE_IMPLEMENTATION GATE_TYPE GATE_DELAY GATE_AREA
// Delays in nanoseconds
// Ignore blank lines, lines with spaces only, and lines starting with "//"

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
XOR2_1 XOR2 2.2 5.9
XOR2_2 XOR2 5.2 2.3
XOR2_3 XOR2 1.2 15.6
```

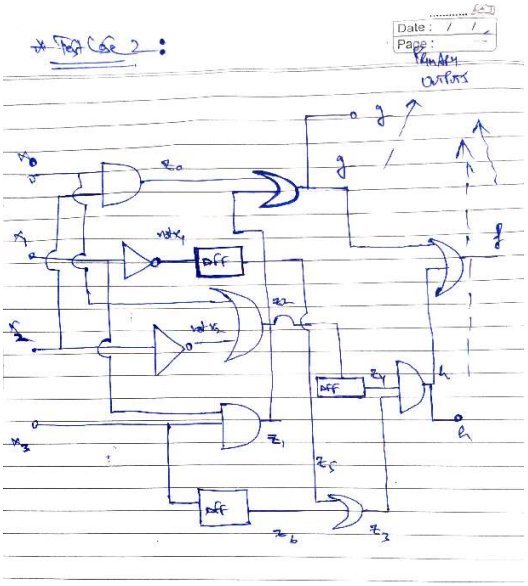
Output:

longest_delay

File Edit View

```
18
```

Test Case2:



circuit

File Edit View

```
PRIMARY_INPUTS x0 x1 x2 x3
INTERNAL_SIGNALS z0 z1 z2 z3 z4 z5 z6
PRIMARY_OUTPUTS g f h
INV x2 notx2
INV x1 notx1
AND2 x0 x2 z0
AND2 x1 x3 z1
DFF notx1 z5
DFF z2 z4
OR2 x0 notx2 z2
OR2 notx1 x3 z3
OR2 z0 z1 g
DFF x3 z6
AND2 z2 z3 h
OR2 g h f
```

gate_delays

File Edit View

```
// Format:
// GATE_IMPLEMENTATION GATE_TYPE GATE_DELAY GATE_AREA
// Delays in nanoseconds
// Ignore blank lines, lines with spaces only, and lines starting with "//"

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
XOR2_1 XOR2 2.2 5.9
XOR2_2 XOR2 5.2 2.3
XOR2_3 XOR2 1.2 15.6
```

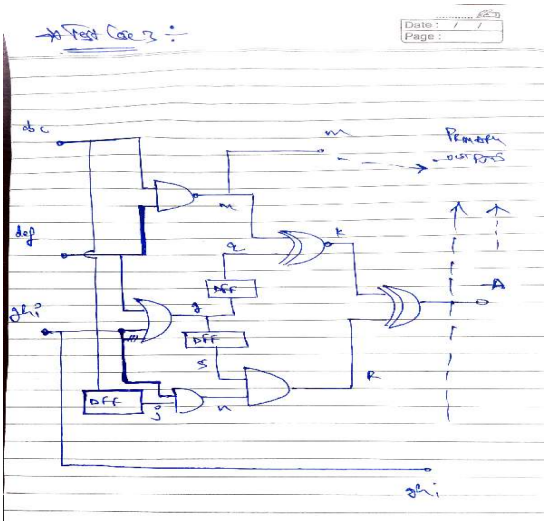
Output:

longest_delay

File Edit View

```
13.5
```

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 j n
DFF g q
DFF g s
AND2 s n R
XNOR2 m q k
XOR2 k R A
DFF abc j
```

gate_delays

File Edit View

```
// Format:
// GATE_IMPLEMENTATION GATE_TYPE GATE_DELAY GATE_AREA
// Delays in nanoseconds
// Ignore blank lines, lines with spaces only, and lines starting with "//"

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
XOR2_1 XOR2 2.2 5.9
XOR2_2 XOR2 5.2 2.3
XOR2_3 XOR2 1.2 15.6
```

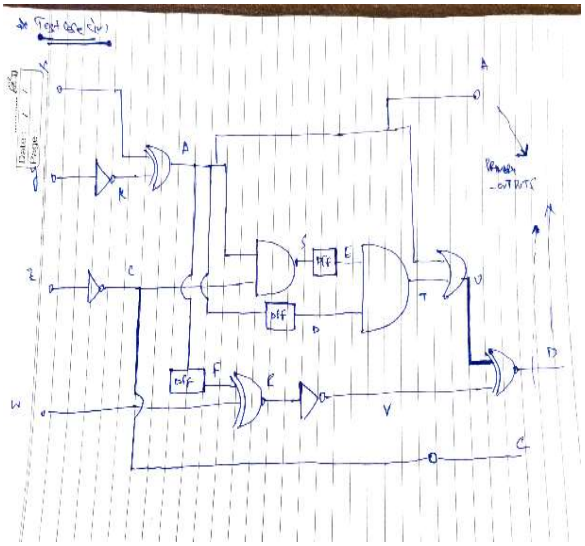
Output:

longest_delay

File Edit View

8

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
```

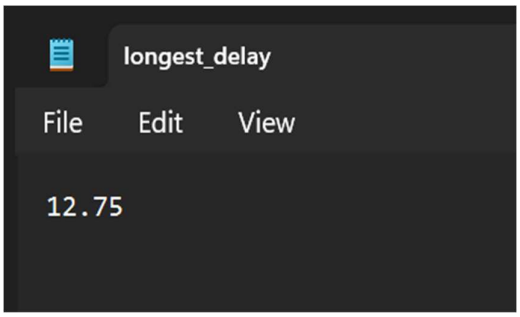
gate_delays

File Edit View

```
// Format:
// GATE_IMPLEMENTATION GATE_TYPE GATE_DELAY GATE_AREA
// Delays in nanoseconds
// Ignore blank lines, lines with spaces only, and lines starting with "//"

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
XOR2_1 XOR2 2.2 5.9
XOR2_2 XOR2 5.2 2.3
XOR2_3 XOR2 1.2 15.6
```

Output:



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 signals 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.