RISHIK SAHA-2022CS11932| PARTH VERMA-2022CS11936

COL215-SOFTWARE_2

REPORT

PART-B

ALGORITHM:

Here we utilize the data stored in the “data” dictionary (as developed in part_a). We then assign the delay_constraint in the form of required delays to the circuit outputs (primary outputs and the signal going into D Flip Flops)

Initially we assign all gates with their fastest implementation (that is the implementation with maximum area and then aim to minimize the total area)

We use the function “calc_delayB” which was developed in part_b of Software Assignment 1, The function utilizes the forward connected graph representation of the circuit and recursively evaluates the maximum time till which each signal in the circuit should be ready for the outputs (primary outputs and

signals going into D Flip Flops) to be available at delay constraint time. This particular maximal time for each signal is maintained by a map “threshold”.

Post this operation, we call a function called “Calc_delayA_new” which utilizes the backward directed graph representation of the circuit.

The function starts its control flow from the inputs (primary inputs and the signal coming out from a DFF) and via DFS visits all the signals.

At a signal the function searches for the implementation of the corresponding gate with minimum area while ensuring that it does not cause its input delay to exceed threshold. This best fit implementation of the gate (corresponding to the signal) is assigned to it and the flow of the function moves forward thereby giving all gates their best possible implementation (that is one with minimum area) and simultaneously respecting the delay_constraint. The function is called on each output signals so that we cover the whole circuit.

Further we traverse over all the gates summing up their areas and finally we print the minimized area in “minimum_area.txt” file.

TIME COMPLEXITY:

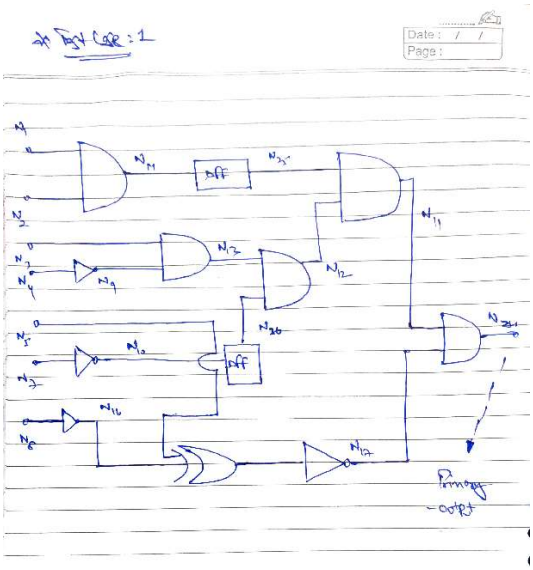
The time complexity of the above algorithm will be

O(E+V) where E and V are the no of edges and vertices in 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
```

delay_constraint

File Edit View

9

Output:

longest_delay

File Edit View

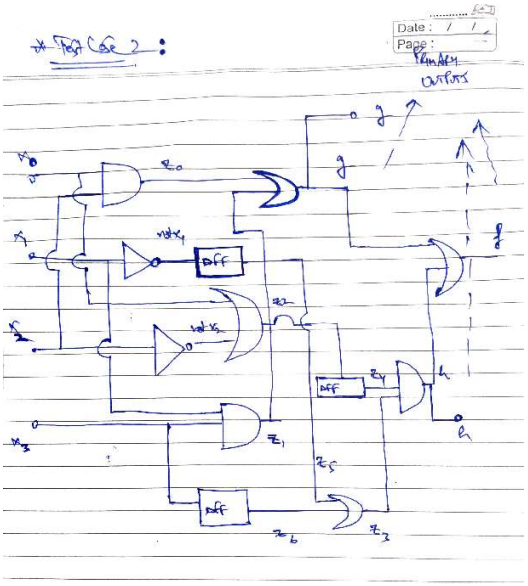
18

minimum_area

File Edit View

130.49

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
```

delay_constraint

File Edit View

9

Output:

longest_delay

File Edit View

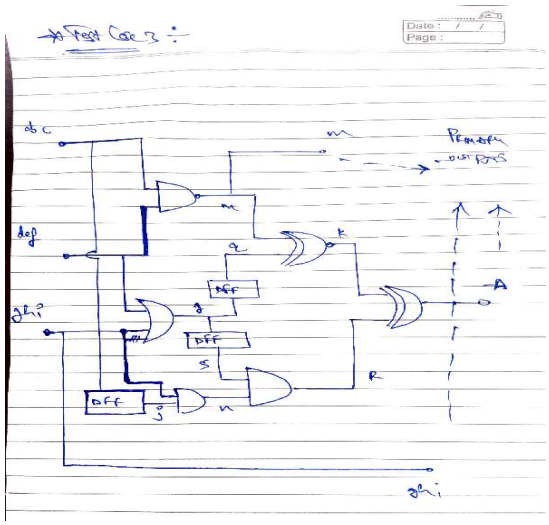
```
13.5
```

minimum_area

File Edit View

```
110
```

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

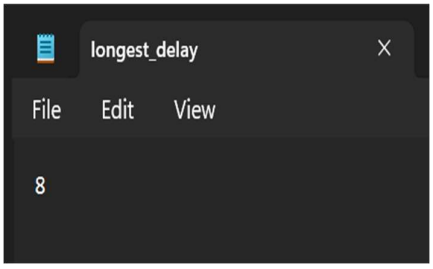
```
// Format:
// GATE_IMPLEMENTATION GATE_TYPE GATE_DELAY GATE_AREA
// Delays in nanoseconds
// Ignore blank lines, lines with spaces only, and lines starting with "//"

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
```

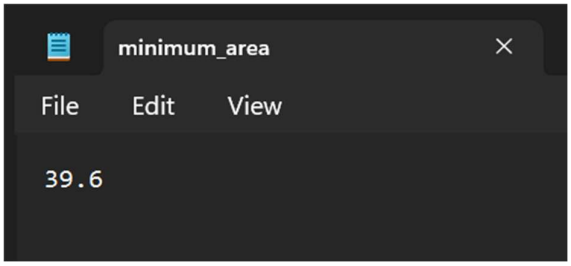
delay_constraint

File Edit View

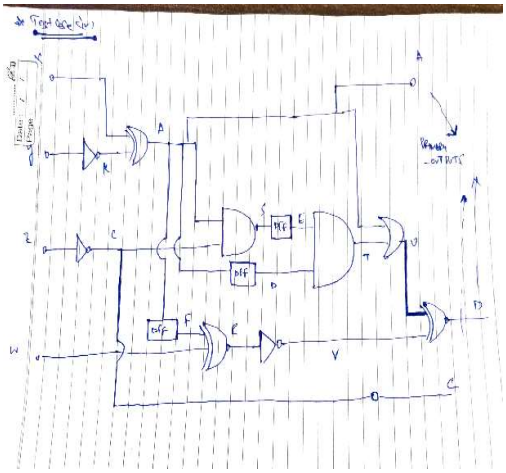
```
20
```

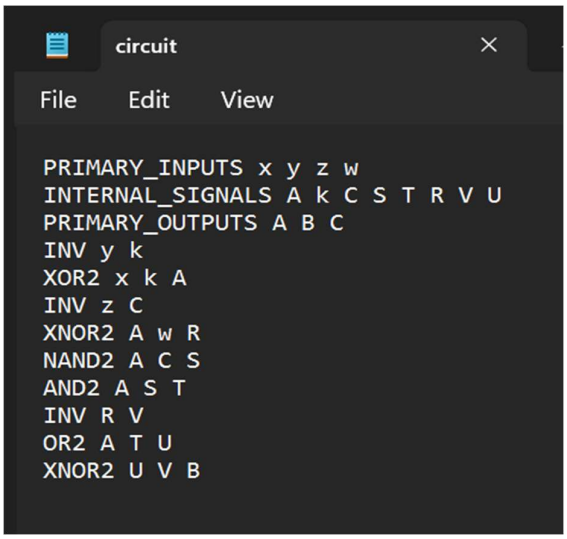
Output:

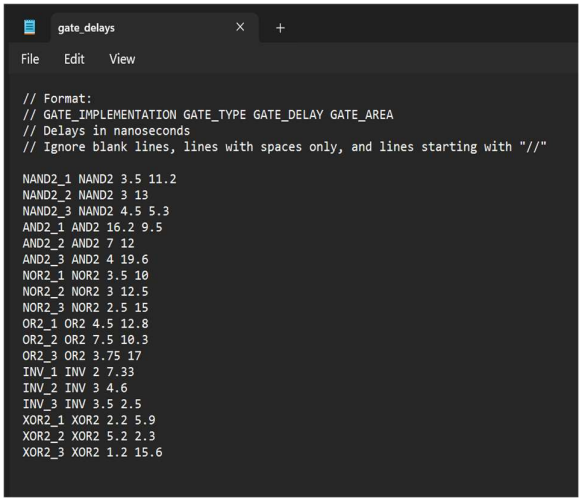


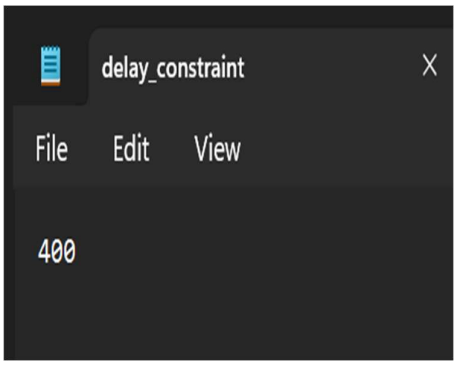


Test Case _4:

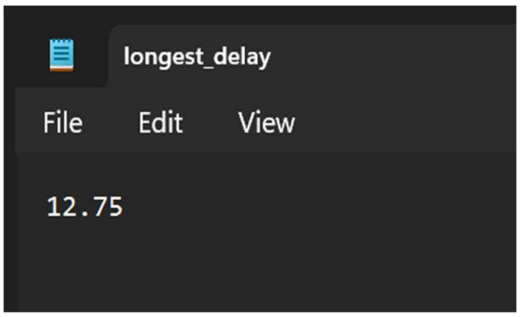


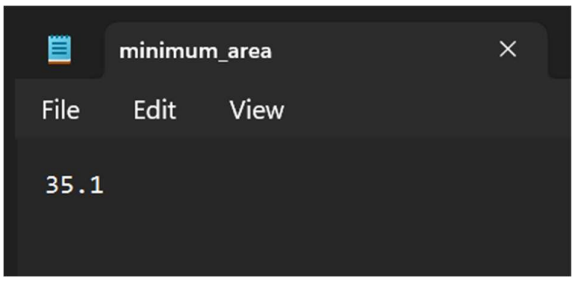






Output:





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.