COL215 Software Assignment 3: Gate Packing

Vanshika (2023CS10746)
Laksh Goel (2023CS10848)

October 20, 2024

1 Task

We are given gates, which contain pins. Gates were assumed to be rectangular and could not be re-oriented. The objective is to minimize the total time delay of the circuit.

2 Design Decisions

We began by extending our software 2 assignment.

2.1 Implementation 1

1. Firstly, we found out how to calculate the maximum delay from all primary input to primary output paths . For that we stored connections of each pin to each of the other pins. Then we found out the primary input and primary output using the fact that they are the pins which never appeared in the wire connections. Now, we start traversing from primary input and discover all the possibilities till we end up on primary output
2. But the problem with this was that there were very high number of possible paths even for less number of gates around 20 and 200 pins. So, storing all the paths and then calculating delay along all the paths was cumbersome and leads to huge amount of time.
3. But another advantage of this was that we could build up our base case of annealing using the paths. How ? We now know that what are all the gates coming in the any one of the paths , we group them together and place them close.

2.2 Implementation 2

1. To reduce the time , we constraint our code from computing same delays again and again using dynamic programming.

2. For this we store all the delays from one pin to any primary output pin using dp whenever we come to any such pin whose max_delay has already been calculated then we don't compute it again.

3. Now, the decision making depends upon time vs better initial placement of annealing. For this we tried various test cases generated and came to conclusion that time and space may grow infinite large for the previous implementation. So , we ended up using the second approach.

2.3 Final Implementation

1. We made a connectivity list that is the list which has connected gates adjacent to each other.

2. Next, we splitted this list into smaller clusters of size 10.

3. The gates in each cluster are arranged tightly using the minimum bounding box concept from the previous assignment.

4. The clusters are also then arranged using the minimum bounding box concept with respect to each other.

5. Then, we apply simulated annealing on clusters.

6. Further, we apply simulated annealing on gates as a whole without considering any clusters.

7. We find the primary inputs and primary outputs as described above and, start traversing using the connectivity list and return the max_delay when we end up getting primary output.

8. After this, we check if there are any cycles.

Note: The main reason for applying simulated annealing on clusters is because when we were not doing so while applying simulated annealing to individual gates, the gates were automatically forming clusters in 2-D plane in the initial simulations and once, they form such clusters the gates won't move from those positions, since the wire length would increase if I move one gate. I need to move the whole cluster, including that gate, for optimisation.

2.3.1 Time complexity analysis

Let, number of gates = g
number of pins total = p
number of wire_connections = w
Time taken by parse_input = O(g+p+w)
Time taken by update_graph function = O(p + w)
Initial_placement = O(g^2) Time in optimising placements using annealing:
For every iteration, we choose a random gate, which takes log(g) amount of time, and we

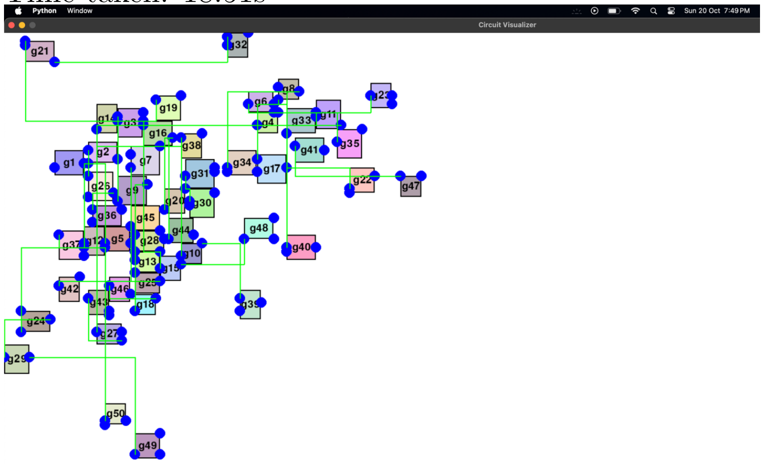
check the overlap for the perturbation and do other updations in coordinates of gates, etc., which takes O(g) amount of time, where g is the number of gates. Calculate_max_delay for given placement takes O (p + w) amount of time. So, finally,
time complexity = O(number_of_iterations * (g + log(g) + p+w)) The time complexity for overall code = summation over all the above-mentioned complexities.

3 Test Cases

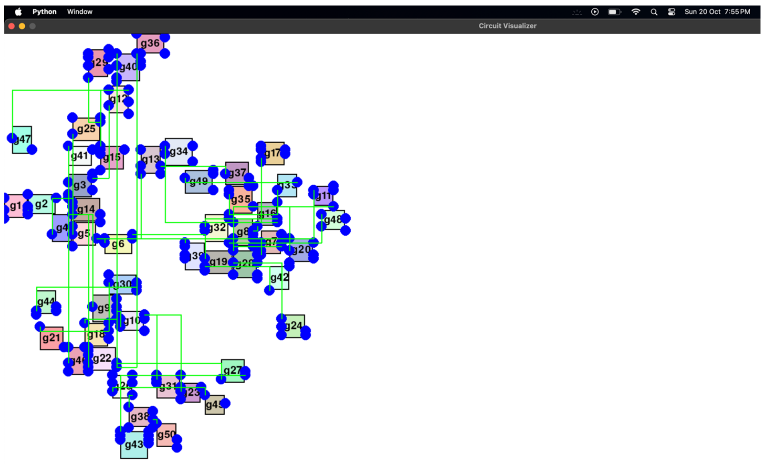
3.1 Test Cases for comparing time with number of pins

Note: These inputs and outputs are submitted as input files; the corresponding names are in brackets

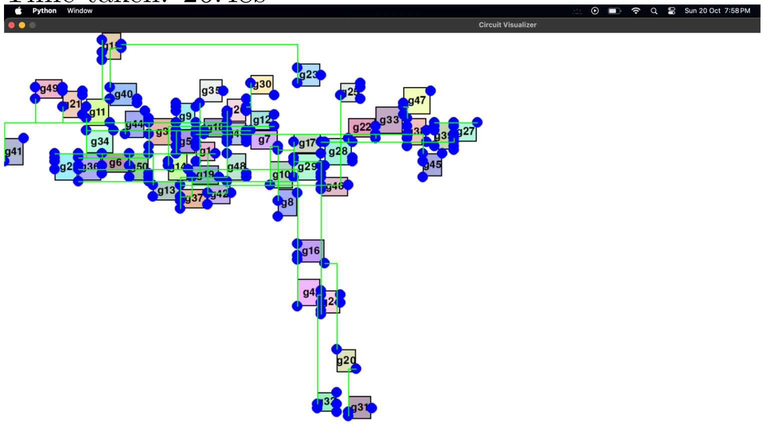
1. Test Case 1(ttc1)
   Gates: 50
   Pins: 200
   Time taken: 18.51s



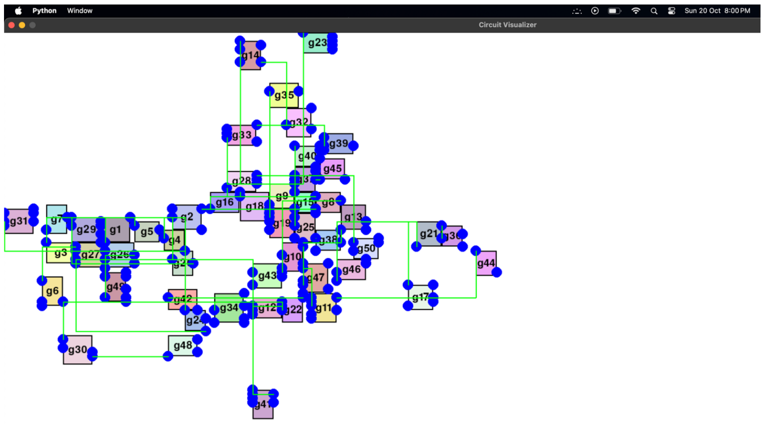
2. Test Case 2(ttc2)
   Gates: 50
   Pins: 500
   Time taken: 35.75s



3. Test Case 3(ttc3)
   Gates: 50
   Pins: 1000
   Time taken: 26.48s



4. Test Case 4(ttc4)
   Gates: 50
   Pins: 2000
   Time taken: 26.03s



3.2 Test Cases which compare between different parameters namely: Number of wire connections per gate (Wire density) , Non-uniformity in sizes of gates(Variance sizes), Non-uniformity in delays(Variance delay)

1. Test Case 1 (ntc1 5 lhl d5)
   Gates: 5
   variance_sizes = 'low'
   wire_density = 'high'
   variance_delays = 'low'

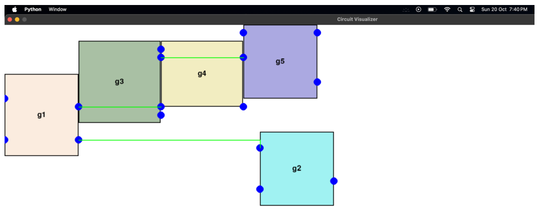


Figure 1: Our code output for above test case

2. Test Case 2(ntc1 10 lhl d5)
   Gates: 10
   variance_sizes = 'low'
   wire_density = 'high'
   variance_delays = 'low'

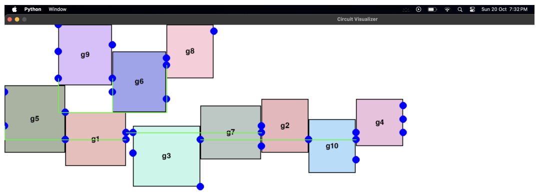


Figure 2: Our code output for above test case

3. Test Case 3 (ntc 1 20 lhl d5)
   Gates: 20
   variance_sizes = 'low'
   wire_density = 'high'
   variance_delays = 'low'

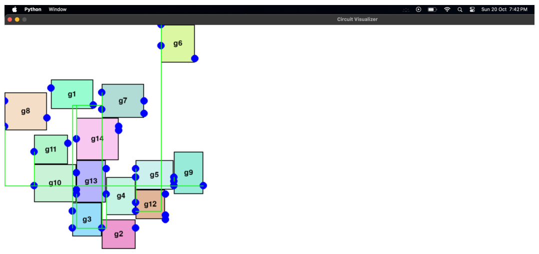


Figure 3: Our code output for above test case

4. Test Case 4 (ntc2 5 llh d5)
   Gates: 5
   variance_sizes = 'low'
   wire_density = 'low'
   variance_delays = 'high'

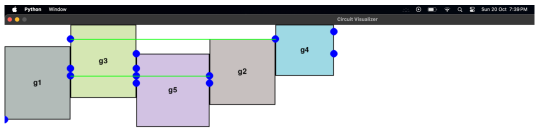


Figure 4: Our code output for above test case

5. Test Case 5 (ntc2 10 llh d5)
   Gates: 10
   variance_sizes = 'low'
   wire_density = 'low'
   variance_delays = 'high'

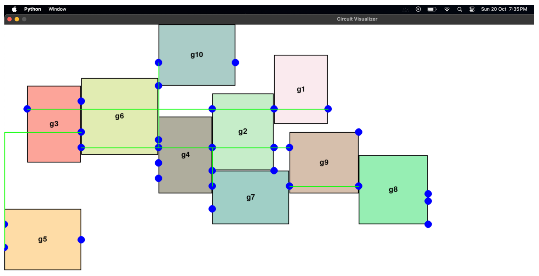


Figure 5: Our code output for above test case

6. Test Case 6 (ntc2 20 llh d5)
   Gates: 20
   variance_sizes = 'low'
   wire_density = 'low'
   variance_delays = 'high'

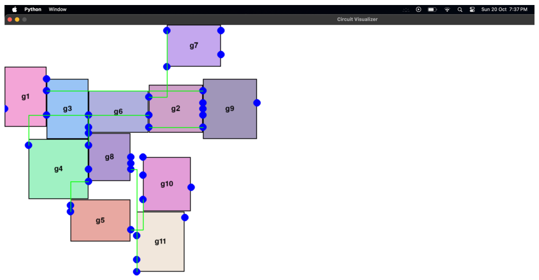


Figure 6: Our code output for above test case

d

3.3 Special Test Cases

1. Test Case 1

Example Test Case:	Output:
g1 30 2 10	bounding_box 60 10
pins g1 0 0 30 2	critical_path g1.p1 g1.p2 g2.p1 g2.p2
g2 30 10 10	critical_path_delay 20
pins g2 0 2 30 20	g1 0 0
wire_delay 1000	g2 30 0
wire g1.p2 g2.p1	

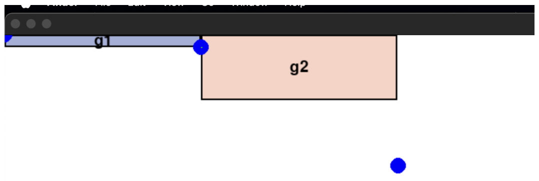


Figure 7: Our code output for above test case

Justification : The above test case was to test whether the code goes against the initial placement strategy to minimise the delay or not and it shows that it does. the only delay is due to gate delays.

2. Test Case 2

Example Test Case:

```
g1 4 4 5
pins g1 0 0 4 2
g2 4 2 5
pins g2 0 1 4 1
g3 4 4 5
pins g3 0 2 4 2
g4 4 2 5
pins g4 0 1 4 1
wire_delay 4
wire g1.p2 g2.p1
wire g2.p2 g3.p1
wire g3.p2 g4.p1
```

Output:

```
bounding_box 16 4
critical_path g1.p1 g1.p2 g2.p1
g2.p2 g3.p1 g3.p2 g4.p1 g4.p2
critical_path_delay 20
g1 0 0
g2 4 1
g3 8 0
g4 12 1
```

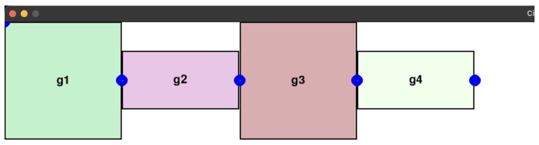


Figure 8: Our code output for above test case

Justification : The above test was to to test our code if it takes a weird gate arrangement to minimise the delay or not and it shows that it does.

3. Test Case 3

Example Test Case:

```
g1 2 2 5
pins g1 0 0 2 1
g2 2 2 5
pins g2 0 1 2 2
g3 2 2 5
pins g3 0 1 2 2
wire_delay 5
wire g1.p2 g2.p1
wire g1.p2 g3.p1
```

Output:

```
bounding_box 6 2
critical_path g1.p1 g1.p2 g3.p1
g3.p2
critical_path_delay 20
g1 0 0
g3 2 0
g2 4 0
```

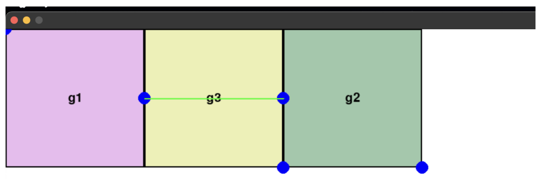


Figure 9: Our code output for above test case

Justification : In this test case we connected output pin to 2 different gate's inputs and expected that semi-perimeter must be minimised by our code because gate-delays summation was fixed here. The most optimal semi-perimeter could be 2 and it is 2 only as shown in the figure.

4. Test Case 4

Example Test Case:	Output:
g1 4 4 5	bounding_box 104 54
pins g1 0 0 4 2	critical_path g1.p1 g1.p2 g2.p1 g2.p2
g2 100 50 100	critical_path_delay 105
pins g2 0 0 100 50	g2 4 4
g3 4 4 5	g1 0 2
pins g3 0 4 4 0	g3 4 0
wire_delay 1000	
wire g1.p2 g2.p1	
wire g1.p2 g3.p1	

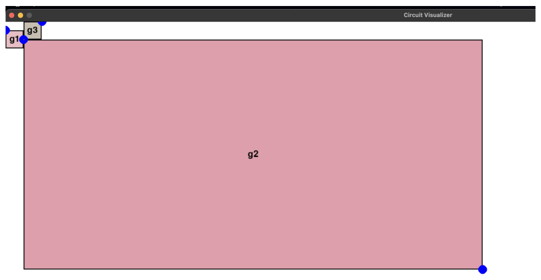


Figure 10: Our code output for above test case

Justification: Here the three pins involved in 2 wire connections could all possibly be overlapped, so we checked if our code was able to do that, despite of the non-uniform geometries of the given gates.This is the most optimal solution where the wire-length involved is ZERO !!

5. Test Case 5

Example Test Case:

```
g1 4 4 5
pins g1 0 0 4 0 4 4
g2 4 4 5
pins g2 0 0 4 0
g3 4 4 500
pins g3 0 4 4 4
wire_delay 50
wire g1.p2 g2.p1
wire g1.p3 g3.p1
```

Output:

```
bounding_box 12 4
critical_path  g1.p1  g1.p3  g3.p1
g3.p2
critical_path_delay 505
g1 0 0
g3 4 0
g2 8 0
```

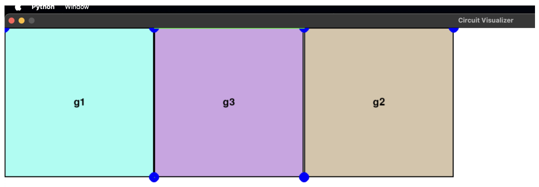


Figure 11: Our code output for above test case

Justification : Here we had put one gate which has primary input , and this gate was connected two other gates, one with very high delay and another one with very low delay. Now, delay along the path which taked low delay gate is already less than the other path. So, we should try to minimise the max delay along all paths , and thus try to minimise the wire length along the path where high-delay gate is present and our code does exactly this !!

6. Test Case 6

Example Test Case:

```
g1 4 4 100
pins g1 0 2 4 2
g2 4 4 5
pins g2 0 2 4 2
g3 4 4 5
pins g3 0 2 4 2
wire_delay 5
wire g1.p2 g2.p1
wire g3.p2 g2.p1
```

Output:

```
bounding_box 12 4
critical_path  g1.p1  g1.p2  g2.p1
g2.p2
critical_path_delay 105
g1 0 0
g2 4 0
g3 8 0
```

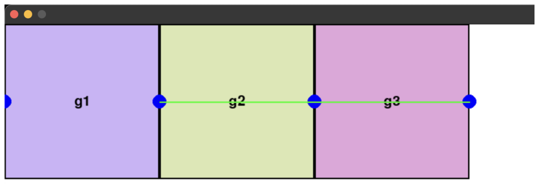


Figure 12: Our code output for above test case

Justification : Here we have primary inputs on two gates g1 and g3, but g1 has very high gate delay and g3 has low gate delay, so g1 should be placed close to g2, as shown in the figure also , which justifies that our code chooses the most optimal solution .

7. Test Case 7

Example Test Case:
```
g1 4 4 100
pins g1 0 2 4 2
g2 4 4 5
pins g2 0 2 4 2
g3 4 4 5
pins g3 0 2 4 2
g4 4 4 10
pins g4 0 2 4 2
wire_delay 5
wire g1.p2 g4.p1
wire g2.p2 g4.p1
wire g3.p2 g4.p1
```

Output:
```
bounding_box 16 4
critical_path g1.p1 g1.p2 g4.p1
g4.p2
critical_path_delay 110
g1 0 0
g4 4 0
g2 8 0
g3 12 0
```

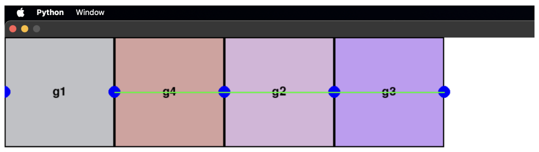


Figure 13: Our code output for above test case

Justification : Here also, we check for the case similar to previous case but with more number of gates.

8. Test Case 8

Example Test Case:

```
g1 4 4 100
pins g1 0 2 4 2
g2 4 4 4
pins g2 0 2 4 2
g3 4 4 4
pins g3 0 2 4 2
g4 4 4 10
pins g4 0 2 4 2
wire_delay 24
wire g1.p2 g4.p1
wire g2.p2 g4.p1
wire g3.p2 g4.p1
```

Output:

```
bounding_box 12 8
critical_path g1.p1 g1.p2 g4.p1
g4.p2
critical_path_delay 110
g1 4 0
g4 8 0
g2 0 0
g3 4 4
```

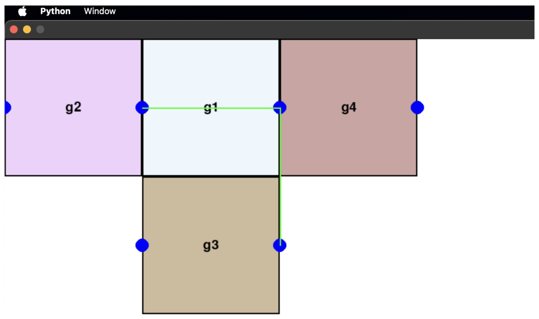


Figure 14: Our code output for above test case

Justification : The above test case is a very strong test case which represents that there is no other way but to minimise all the possible paths to same delay of 110, increasing delay by even 1 makes it as the longest delay path.So, this is the most optimal one.

9. Test Case 9

Example Test Case:

```
g1 20 10 5
pins g1 0 6 20 10
g2 20 30 10
pins g2 0 7 20 4
g3 15 15 10
pins g3 0 7 15 15
wire_delay 7
wire g1.p2 g2.p1
wire g2.p2 g3.p1
```

Output:

```
bounding_box 55 33
critical_path g1.p1 g1.p2 g2.p1
g2.p2 g3.p1 g3.p2
critical_path_delay 25
g2 20 3
g1 0 0
g3 40 0
```

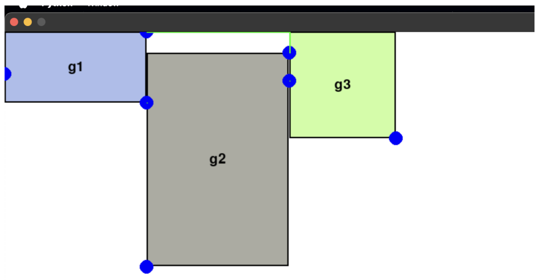


Figure 15: Our code output for above test case

Justification: Since Wire lengths are zero, therefore delay is sum of delay of gates, which is the minimum possible answer possible in this scenario and hence the most optimal.

10. Test Case 10

Example Test Case:
g1 20 10 5
pins g1 0 6 20 10 20 0
g2 20 30 100
pins g2 0 7 20 4 20 0
g3 15 15 10
pins g3 0 7 15 15 0 3
wire_delay 1
wire g1.p2 g2.p1
wire g2.p2 g3.p1
wire g1.p3 g3.p3
wire g2.p3 g3.p3

Output:
bounding_box 55 33
critical_path g1.p1 g1.p2 g2.p1 g2.p2 g3.p1 g3.p2
critical_path_delay 115
g2 20 3
g1 0 0
g3 40 0

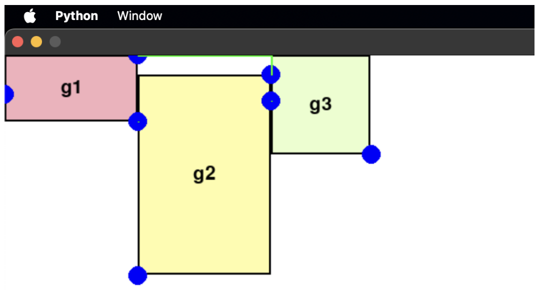


Figure 16: Our code output for above test case

Justification : It minimises the wire length along the path where the high gate delay is present, because it remains to be the highest in any case we must optimise that.

11. Test Case 11

Example Test Case:
g1 20 10 100
pins g1 0 6 20 10 20 0
g2 20 30 10
pins g2 0 7 20 4 20 0 0 30
g3 15 15 10
pins g3 0 7 15 15 0 3
wire_delay 1
wire g1.p2 g2.p1
wire g2.p2 g3.p1
wire g1.p3 g3.p3
wire g2.p3 g3.p3

Output:
bounding_box 55 31
critical_path g1.p1 g1.p3 g3.p3 g3.p2
critical_path_delay 130
g2 20 1
g1 0 3
g3 40 0

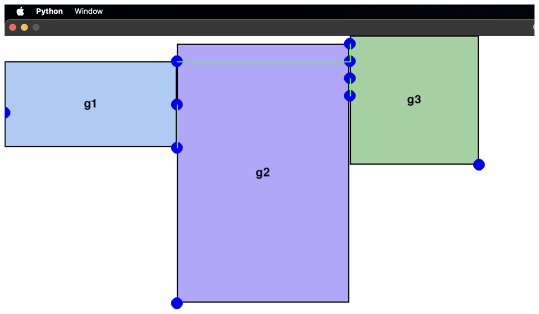


Figure 17: Our code output for above test case

Justification : This is the most optimal strategy , the gate with primary input is of high delay here, which does not have effect, it needs to be counted in any case, so minimise the wire length along all paths.

3.4 Given Test Cases on moodle

1. Test Case 1

Example Test Case:
g1 2 3 5
pins g1 0 1 2 2
g2 3 2 3
pins g2 0 0 3 1
g3 2 2 6
pins g3 0 1 0 2 2 1
wire_delay 4
wire g1.p2 g3.p1
wire g2.p2 g3.p2

Output:
bounding_box 5 5
critical_path g2.p1 g2.p2 g3.p2 g3.p3
critical_path_delay 13
g1 1 0
g2 0 3
g3 3 1

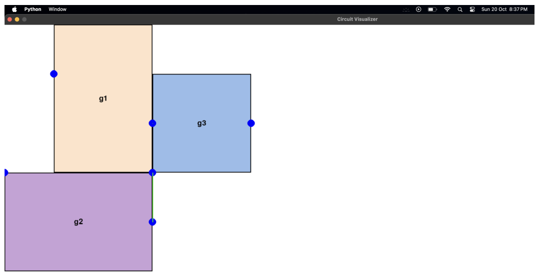


Figure 18: Our code output for above test case

2. Test Case 2

Example Test Case:
```
g1 2 3 5
pins g1 0 1 2 2
g2 3 2 3
pins g2 0 0 3 1
g3 2 2 6
pins g3 0 1 0 2 2 1
g4 2 3 4
pins g4 0 0 0 1 2 2
wire_delay 3
wire g1.p2 g3.p1
wire g2.p2 g3.p2
wire g2.p2 g4.p1
wire g3.p3 g4.p2
```

Output:
```
bounding_box 7 6
critical_path g2.p1 g2.p2 g3.p2
g3.p3 g4.p2 g4.p3
critical_path_delay 25
g1 1 0
g4 5 3
g2 0 3
g3 3 2
```

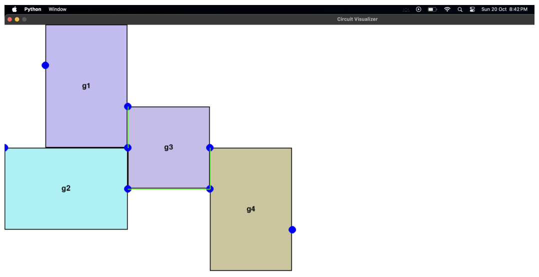


Figure 19: Our code output for above test case

4 Time Analysis

Table 1: Readings for Open Circuit Test

No. of Gates	No. of Pins	Time taken	Input/output File Name	Number of wire connection
50	2000	26.03	tt4	199
200	8000	8.04	time_input2.txt	199
1000	40000	3.31	time_input3.txt	999