COL215: Digital Logic and System Design
Software Assignment - 1: Circuit Delays

Yash Bansal (2022CS51133)
Soumyaprabha Dey (2022CS11107)

1 Algorithm

In this algorithm, we use memorization

Part A: Parsing and Storing of the Circuit:-

Parsing and Storing of the Circuit Specifications

- From the file "`gate_delays.txt`, parse the input and create the dictionary `gate_delays` using the gate names as keys and corresponding delays as values.
- Create three lists: `primary_inputs`, `primary_outputs`, and `internal_signals`, and four dictionaries: `out_circuit_graph`, `out_generator_gate`, `out_delay_evaluated`, and `out_delay`.
- Read and parse the circuit specifications from the `circuit.txt` file.
- Store inputs, outputs, and signals in lists: `primary_inputs`, `primary_outputs`, and `internal_signals`.
- Store circuit specifications in `out_circuit_graph` and `out_generator_gate` dictionaries:
  - `out_circuit_graph` stores signals as keys and a tuple of corresponding generating signals as values.
  - `out_generator_gate` stores signals as keys and corresponding generating gate as values.
  - For example, if signal `C` is generated from signals `A` and `B` by a NAND2 gate, then (`C`, (`A`, `B`)) is a key-value pair of `out_circuit_graph`.
  - (`C`, `NAND2`) is a key-value pair of `out_generator_gate`.
  - Initially, `out_delay_evaluated` is `true` for inputs and `false` otherwise.
  - Initially, `out_delay` is 0 for inputs and `None` otherwise.

Function to Find Output Delays

The function `find_out_delay()` is called on every output, and the results are stored in a list `res`. When `find_out_delay(x)` is called:

- If the value of `out_delay[x]` is already evaluated, return it (in constant time).
- If not, find the maximum delay of child nodes (`y`) and add it to the delay of the gate that generates signal `x`.
- Store the value in `out_delay[x]` and mark that `out_delay[x]` has been evaluated.
- Return `out_delay[x]`.

Part B: Parsing and Storing of the Circuit:-

Parsing and Storing of the Circuit Specifications

To find input delays, modify the directed `out_circuit_graph` to `in_circuit_graph`:

- If signal `A` was a parent of signal `B` in `out_circuit_graph`, then signal `B` will be a parent of signal `A` in `in_circuit_graph`.
- For example, if signal `A` was generated from signals `B` and `C` by a NAND2 gate, then `B` and `C` are parents of `A`.
- Information about the NAND2 gate is stored in `in_generator_gate` with keys `B` and `C` as values.
- `in_circuit_graph` and `in_generator_gate` only have inputs and signals as keys.
- `in_delay_evaluated` contains information on whether an input, signal, or output delay has been evaluated.
- `in_delay` stores the delay of inputs, signals, and outputs.
- Read input from `required_delays.txt` and store delays of outputs in `in_delay`.
- Initially, `in_delay_evaluated` is `true` for outputs and `false` otherwise.
- Initially, `in_delay` is present for outputs and `None` otherwise.

Function to Find Input Delays

The function `find_in_delay()` is called on every input, and the results are stored in a list `res`. When `find_in_delay(x)` is called:

- If `in_delay[x]` has been evaluated before, return it in constant time.
- Compute the delay possible in signal X as the minimum{ delay(y) - delay of gate which generates y from x — y is a child node of x}.
- This value is stored in `in_delay[x]`, and `in_delay_evaluated[x]` is set to `true`.
- Return `in_delay[x]`.

Out_circuit_graph and in_circuit_graph are directed graphs stored as dictionary data structures.

2 Correctness

Part A:-

Lemma

If there are n nodes in a directed graph without any cycles, then there exists a node with degree 0.

Proof

We use a proof by contradiction. Suppose all nodes have degree greater than 0. Take a node N_1. As it has a degree greater than 0, it is connected to another node N_2 with a degree greater than 0. N_2 will be connected to another node N_3, which is not N_1 (as the graph has no cycles). This will continue indefinitely. The number of such connections can't be more than ${}^nC_2$ for a graph with n nodes, leading to a contradiction. Hence, there must be a node with degree 0.

In the context of our algorithm, `out_circuit_graph` is a directed graph. The outputs correspond to those nodes which have a degree of 0.

Now, we prove the correctness of the algorithm using induction.

Induction Hypothesis

Let $P(n)$ be the proposition that `find_out_delay(x)` computes the delays of all nodes of `out_circuit_graph` with $n \geq 2$ nodes, with at least one input and one output, correctly.

Induction Base: $n = 2$

In this case, there is exactly one input and one output, and no internal signals.

- So, `find_out_delay(x)`, by definition, finds the output delay of inputs correctly. The delay of the output is the delay of the input plus the delay of the gate.
- Calling `find_out_delay` on the output evaluates it correctly.

Induction Step: $P(n) \Rightarrow P(n+1)$

Suppose $P(n)$ holds for some $n \geq 2$.

Assume P(n) holds for some $n \geq 2$. Consider a out_circuit_graph with (n+1) nodes with atleast one input and output. By definition, find_out_delay() computes the delays of all inputs(=0) correctly. Seperate one output, X from the garph. The delays of the remaining n nodes don't depend on the output X, by lemma, it has one output, and obviously it also has a input, and are evaluated correctly by find_out_delay(), according to the induction hypothesis.

Now rejoin the seperated node X. Its delay is the maximum of the child node delays plus delay of the generating gate, as computed by find_out_delay(X).

Hnece, proved.

Part B:-

The `in_circuit_graph` is the just reverse of the directed `out_circuit_graph`, and find_out_delay() is replaced by find_in_delay(), ensuring that find_in_delay() computes in delay of all signals correctly.

3 Time Complexity

Part A:-

The complexity of our algorithm is O(number of edges of the graph).

Proof

Consider, in general, any node X in out_circuit_graph. If out_delay[X] s already evaluated, then we just return it. If not, iterate over all the adjacent vertices $\{Y_1\ ,\ Y_2\ ...\ Y_n\}$ of X. We need to do degree(X) such iterations, which will cost degree(X). Then, we do the same for $\{Y_1\ ,\ Y_2\ ...\ Y_n\}$. Let $\{O_1\ ,\ O_2\ ...\ O_k\}$ be the set of outputs. If we call find_out_delay() on elements of this set, all the edges of the out_circuit_graph will be traversed and since, we are using memorization, out algorithm traverses every edge exactly once. Hence, total cost is number of edges.

Hence, the complexity of our algorithm is O(Number of edges in the graph).

Part B:-

The `in_circuit_graph` is the just reverse of the directed `out_circuit_graph`. Similarly, the complexity of this algorithm is also O(number of edges in the garph).

4 Testcases

- **Testcase 1:-** This testcase was given in the question as an example.
- **Testcase 2:-** This testcase is to take into account the case when same inputs are given to multiple gates.

- **Testcase 3:-** This testcase takes into account the case when two same gates are placed consecutively.
- **Testcase 4:-** This testcase is to check the working of more than two inputs.
- **Testcase 5:-** This testcase checks when a gate takes two same inputs.
- **Testcase 6:-** This testcase checks the case when some of the gates have zero delays.
- **Testcase 7:-** This testcase checks the case when all of the gates have zero delays.
- **Testcase 8:-** This testcase checks the case when only one input is given.
- **Testcase 9:-** This testcase checks the case when only one output is recieved.
- **Testcase 10:-** This is a complex test case with very complex circuit taking into account multiple situations previously described.