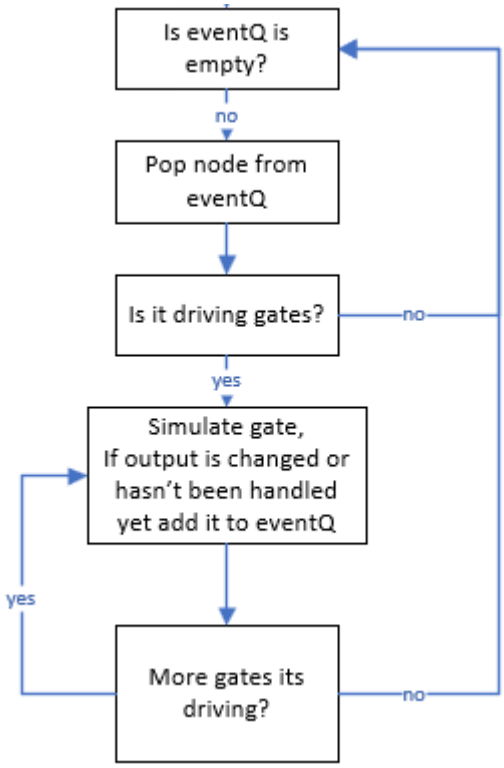
Dry – Questions Part 1:

1. eventQ – A queue of nodes that has been changed.
   nodesMap – A map of all nodes. Each node gets a property of value.
2. 

3. Time is not modeled within a single time unit, the simulator gets the input vector for a specific time unit and updates the node values such that before moving to the next time step (next vector) all the nodes are with the right values. The order of updating the nodes within a single time unit is not necessarily the same as "in real life" but the correctness of the node values by the end of the time unit is promised.
4. Circuits with circles may not be simulated correctly or circuits with multiple drivers to the same node, other than that, no restrictions.

5.

Changing the stdcell.v DFF implementation:
We didn't need to change the algorithm to support the implementation of the flip flop. The feature that enables the algorithm to support it, is that we continue to add the nodes to the eventQ each time a node is changed. So if the loop in the flip flop implementation is changing node values, the gates in the loop will be added to the eventQ until the loop stabilizes and the values are not changing.

Dry – Questions Part 2:

6.1. O(n) – We go over all gates, and each gate can be processed at most as the number of its inputs, which is bounded. We can simulate by ranks, ensuring all the inputs of the gate have been processed. This way each gate will be processed only once (except for loops).

6.2.

- simulate – builds a list of inputs to the gate and simulates the gate by calling simGate. If the output node changed its value or hasn't been handled yet it adds to the eventQ and changes it value.
- simGate – simulates the gate by calling one of:
  - simAnd
  - simOr
  - simNand
  - simNor
  - simXor
  - simBuffer
  - simInv
  - simDff

6.3.

- Adding to the queue:
  - setNodeValue – changing the node value to the new value.
- Removing from the queue:
  - Going over all gets driven from that node and simulating each gate:
  - simulate