Static Timing Analysis (STA) refers to analyzing a circuit graph to find properties such as arrival time, required time and the slack for every gate in the circuit. Use the following notations when writing your answer:

- *G(V,F, E)* is the directed acyclic graph (DAG) representing the circuit, *V* is the set of gates, *F* is the set of flip-flops and *E* is the set of edges *(x,y)*, where $x,y \in V \cup F$. Assume wires (edges) have zero delay.
- *GateType*(v_i) is in the set {gate, flip-flop, input pad, output pad}.
- Fanin(v_i) is the set of gates whose outputs are connected to the inputs of v_i.
- Fanout(v_i) is the set of gates whose inputs are driven by the output of gate v_i.
- Each gate has an intrinsic delay of 1from any input to the output of the gate (both rise time and fall time).
- Setup and hold times for flip flops are zero.

(a) (0.5 points) Assuming that the clock starts at time 0, and given a fixed clock period *T*, provide the equations for calculating the following attributes for a gate "v_i". Your definition should handle all gate types. Explain how you handle flip-flops.

First we break each flip-flop into flip-flop input (ffi) and flip-flop output (ffo) pairs. We can treat ffi as primary output pads, and ffo as primary inputs.

- ArrivalTime (v_i)

  if GateType(vi) ∈ {ffo, input pad},
  ArrivalTime(vi) = 0
  else ArrivalTime(vi) = 1 + max{ArrivalTime(vj)}, where vj ∈ fanin(vi)

- RequiredTime (v_i)

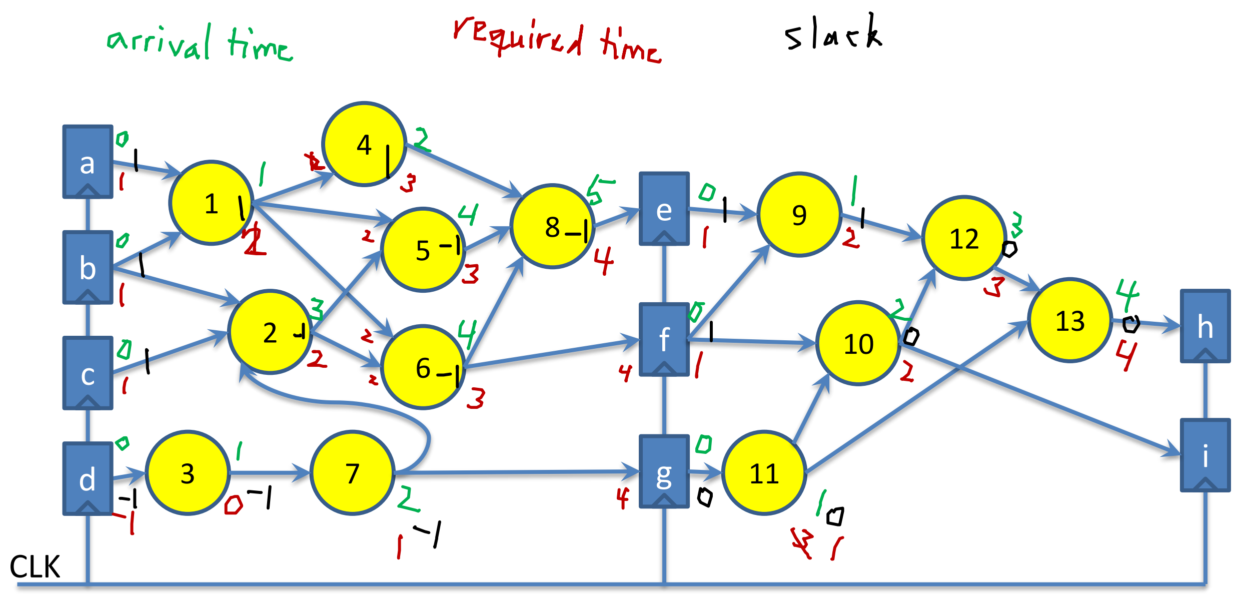
  if GateType(vi) ∈ {ffi, output pad},
  RequiredTime(vi) = T
  else RequiredTime (vi) = min{ RequiredTime (vj)} - 1, where vj ∈ fanout(vi)

- Slack (v_i)

  Slack(vi) = RequiredTime(vi) – ArrivalTime(vi)

(b) (0.7 points) Show the numeric values for the arrival time, required time and slack for the gates in the circuit below. Clock period T=4. Explain how you handle flip-flops when performing timing analysis.

We assume all flip-flops provide their output at time 0, and require their input values to settle before T=4.



(c) (0.3 points) To meet timing constraints in the circuit in part (b), which gates should be sized to make them faster? Explain.

The gates with the smallest slack are on the critical path and they are candidates for timing optimization. The paths 3,7,2,5,8 and 3,7,2,6,8 have negative slack, which means they are violating timing. Any gate on these paths would be a candidate for optimization.

(d) (1.5 points) When multiple gates are candidates for sizing, a tie-breaker can be the number of paths on which a gate is. However, listing all paths explicitly could take exponential runtime.

There is a method for counting the *number of paths* a gate is on, without explicitly listing all such paths. All we need to do is to count the number of incoming paths (i.e., from an input pad or flip-flop output to the gate), and the number of outgoing paths (i.e., paths starting from the output of the gate, leading to an output pad or flip-flop input). Multiplying these two numbers will get us the total number of paths the gate is on. For example, in the circuit of path (b), there are two incoming paths to gate 10 (f →10, g→11→10), and two outgoing paths (10→12→13→h, 10→i). Multiplying these two numbers gets us 4, which is the number of paths that are affected by the delay of gate 10.

Write the pseudo-code for calculating the number of incoming paths and the number of outgoing paths for every gate in the circuit.

The pseudo-code is follows a topological sort traversal once from inputs to outputs to get the number of incoming paths, and then a reverse topological sort traversal to get the number of outgoing paths.

```
FindNumPaths()
{
```

```
set inputs = { xi | xi ∈ $V \cup F$, GateType(xi)=ffo or input pad}
set outputs = { xi | xi ∈ $V \cup F$, GateType(xi)=ffi or output pad}
int numInpPaths[ |V| ], numOutPaths [ |V| ]

initialize numInpPaths[i]=0 and numOutPaths[i]=0, for all vi ∈ V
findNumInpPaths(inputs, numInpPaths)
findNumOutPaths(outputs, numOutPaths)
}

findNumInpPaths (inputs, numInpPaths)
{
set Q=inputs
while Q != ϕ
    Q = Q \ xi
    ∀ xj, xj ∈ fanout(xj) AND GateType(xj) ≠ffo AND GateType(xj) ≠outputPad,
        numInpPaths(j) += numInpPaths(i)
        if xj ∉ Q,
            Q = Q ∪ {xj}
}
```

findNumOutPaths is similar, except we start with outputs, and instead of iterating over the fanouts of xi, we iterate over fanins of xi.

(e) (1 point) This part is unrelated to part (d). In today's VLSI designs it is common to have multiple clock domains. For example, in the circuit of part (b), flip-flops "f" and "h" could run on a clock that is different from the clocks of the rest of the flip-flops in the figure (the circuit would become a 2-clock domain circuit). Assuming there are *k* clock domains, describe how to find the slack of all gates with respect to all clock domains.

Instead of one set of arrival time, required time and slack, we will calculate k-tuple vectors of these values. Each independently calculated based on the clock period and connections to each clock domain. The slack of a gate would be the minimum slack over all slacks in the slack vector.