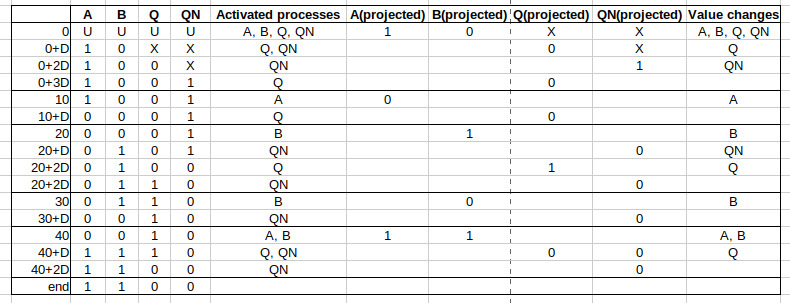
# Task 1

2. At the start of the simulation, all processes are activated.  
In each simulation cycle, the simulator goes to the time of the next event (from the event queue) and updates all the signal that had a scheduled value change.   
It then executes each process that had any of the changed signals in their sensitivity lists and schedules new events in case the execution will lead to new values for the signals affected by the process.   
This is repeated until the event queue is empty or we hit the simulation time limit (set by the user) or in case of oscillations, the simulator times-out.

3. When executing a process causes a signal's value to change, the simulator schedules that value change in the future as a new event in the queue.   
If the change had an explicit delay, the scheduled time for the value change is current time + explicit delay.   
Otherwise, the scheduled time is current time + delta delay. The delta delay can have a duration that depends on each simulator and its implementation.

# Task 2

* Part 1:



* Part 2:

Yes it stops, no oscillation.  
Note: oscillations would happen in the following scenario: A=B=1 🡪 A=B=0.

* Part 3:

|  |  |
| --- | --- |
| -- Testbench  library IEEE;  use IEEE.std\_logic\_1164.all;  entity tb is  end entity tb;  architecture stimuli of tb is  signal A : std\_ulogic;  signal B : std\_ulogic;  signal Q : std\_ulogic;  signal QN : std\_ulogic;  begin  DUT: entity work.assig2 port map (A => A, B => B, Q => Q, QN => QN);  process is  begin  A <= '1' ;  B <= '0' ;  wait for 10 ns ;  A <= '0' ;  wait for 10 ns ;  B <= '1' ;  wait for 10 ns ;  B <= '0' ;  wait for 10 ns ;  B <= '1' ;  A <= '1' ;  wait for 10 ns ;  -- Oscillation here:  -- B <= '0' ;  -- A <= '0' ;  -- wait for 10 ns ;  wait;  end process;  end architecture stimuli; | -- Design  library IEEE;  use IEEE.std\_logic\_1164.all;  entity assig2 is  port (  A : in std\_ulogic;  B : in std\_ulogic;  Q : out std\_ulogic;  QN: out std\_ulogic  );  end entity assig2;  architecture rtl of assig2 is  begin  Q <= A nor QN;  QN <= B nor Q;  end architecture rtl; |