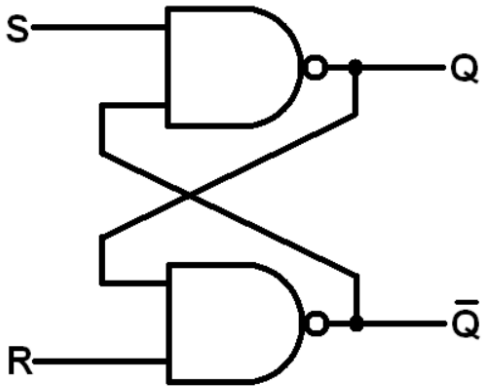
Parisa yahyapour fatideh / 810101551

Computer assignment 4 / ECE 367

Question 1:

Part a)



As we know for SR latch with NAND gates, the truth table will be like below table:

S	R	Q	~Q
0	0	invalid	
0	1	1	0
1	0	0	1
1	1	Q	~Q

Verilog:

```
1   `timescale 1ns/1ns
2   module SRlatch (input s, r ,n,m, output q, qb);
3       nand #12 g1(q,s,qb,n);
4       nand #12 g2(qb,r,q,m);
5   endmodule
6
```

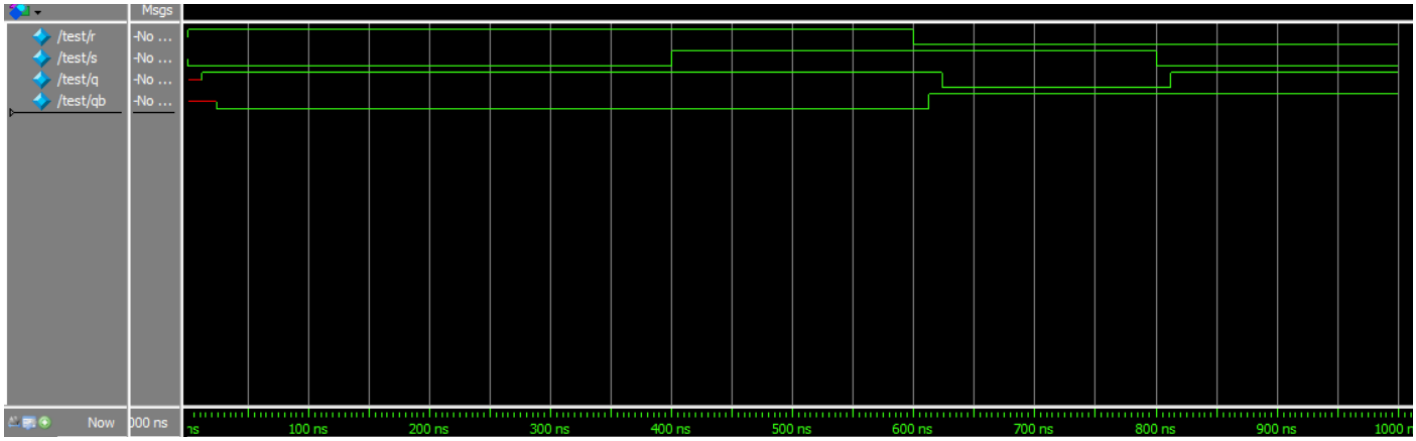
For implementing in a way that NAND gates can be used as 3 or 2 inputs, we use 3-input NAND for both upper and lower gate. When we want to use it as 2-input NAND, we initialize one of the inputs in a way that it doesn't effect on NAND's output which is 1.

Part b)

For calculating delay of gates, switch level is considered. As it's clear, for having 1 on the output of 3-input NAND, NMOSs should put Z on the output which takes 12NS. As there is one delay for transistors so TO1 and TO0 and TOz are the same.

Part c)

Based on the truth table that was shown in part a, in test bench, R and S are changing. The wave form is like below:



At the beginning, r = 1, s = 0. Two inputs that choose the number of inputs which are n and m are equal to 1 because both gates have two inputs.

For 12ns q = X because of gates' delay. As s = 0 so q will be set to 1. After that qb will change after more 12ns because it's related to q.

When s = 1 after 200ns, based on the table, q and qb won't change and keep their previous value.

When r = 0 as s = 1 output will be reset. 12ns after changing r, qb will be 1 because r is directly connected to qb, the NAND gate that is related to qb will be activated. 12ns after changing qb, q will be 0.

As it was in the table, situation that s = r = 0 is invalid. In wave form it can be understood. The way that SR latch is working is as below:

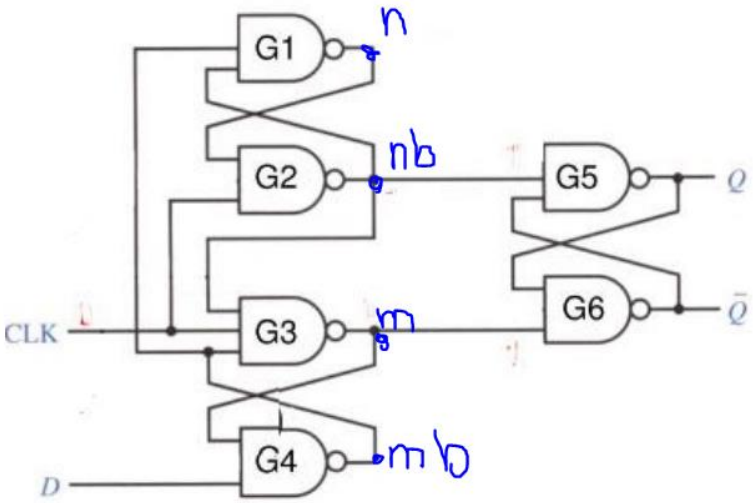
If last negative pulse is on S, then Q = 1.

If last negative pulse is on R, then Q = 0.

Here after setting r and s to 0, both q and qb will be 1, so we can recognize where was the last negative pulse, so the SR latch has lost its memory.

Question 2:

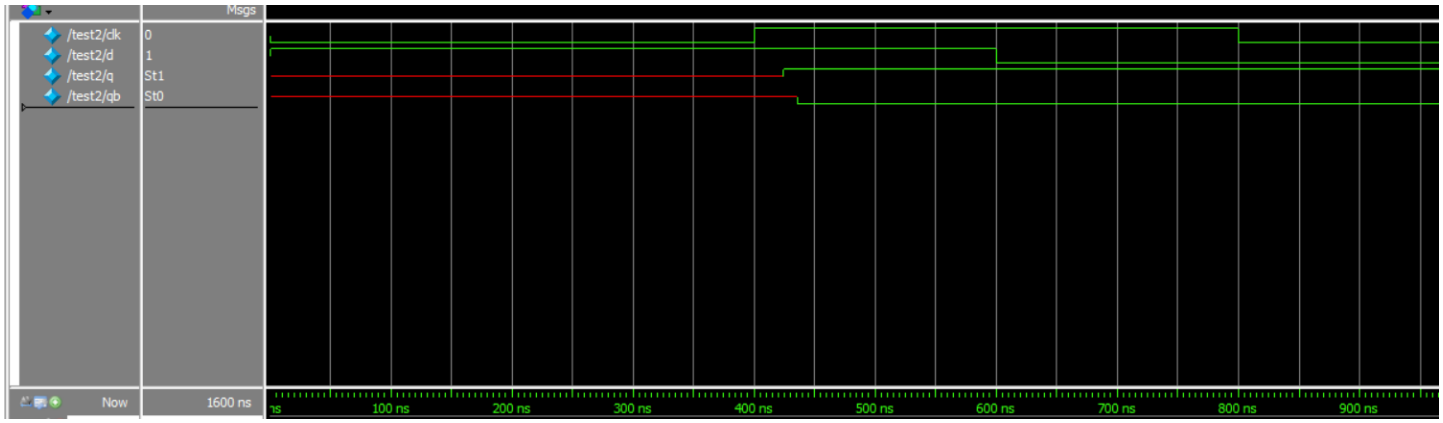
Part a)

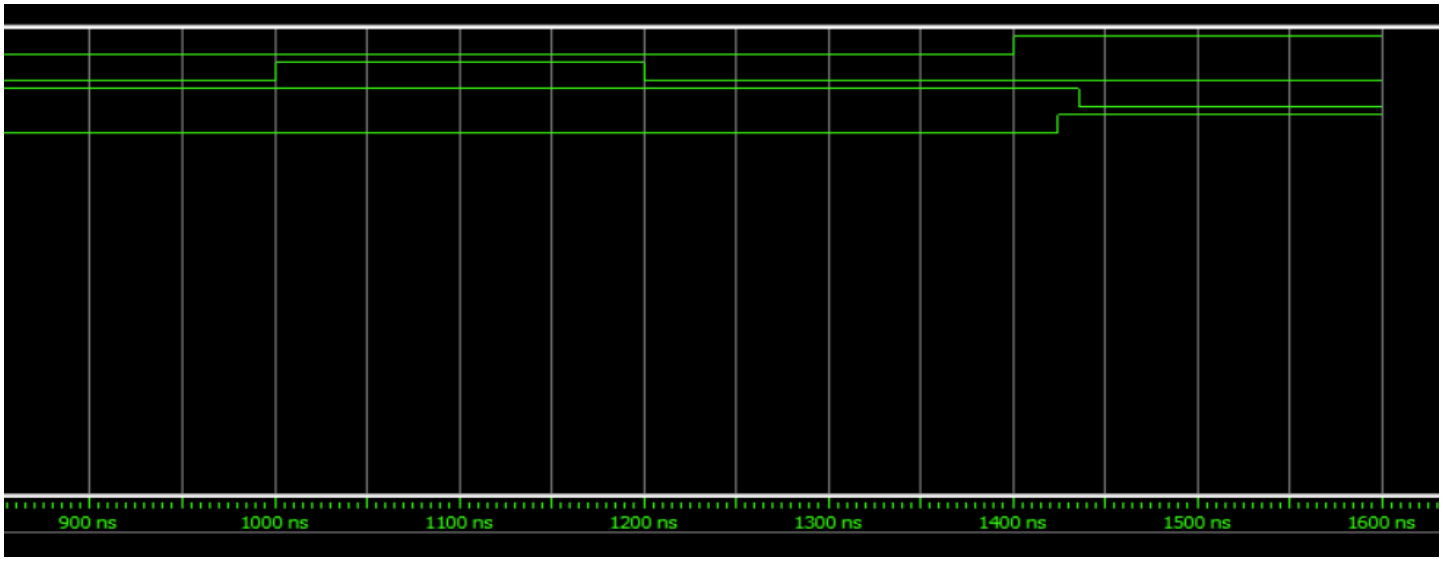


Based on this diagram, Verilog code will be written. As G1, G2 and G4 have 2 inputs, the input of SR latches that choose the number of inputs, for these gates is equal to 1. Verilog code is as below:

```
= q2.v
`timescale 1ns/1ns
module DFF(input clk, d, output q, qb);
    wire n,nb,m,mb;
    SRlatch sr1(mb,clk,1'b1,1'b1,n,nb);
    SRlatch sr2(nb,d,clk,1'b1,m,mb);
    SRlatch sr3(nb,m,1'b1,1'b1,q,qb);
endmodule
```

Part b)





Because of gates' delay, at the beginning we don't know what in on q and qb so till the first posedge clock q and qb are equal to X.

The table of edge trigger d flip flop is as below:

clock	data	Q	~Q
0	0	Q	~Q
0	1	Q	~Q
1	0	0	1
1	1	1	0

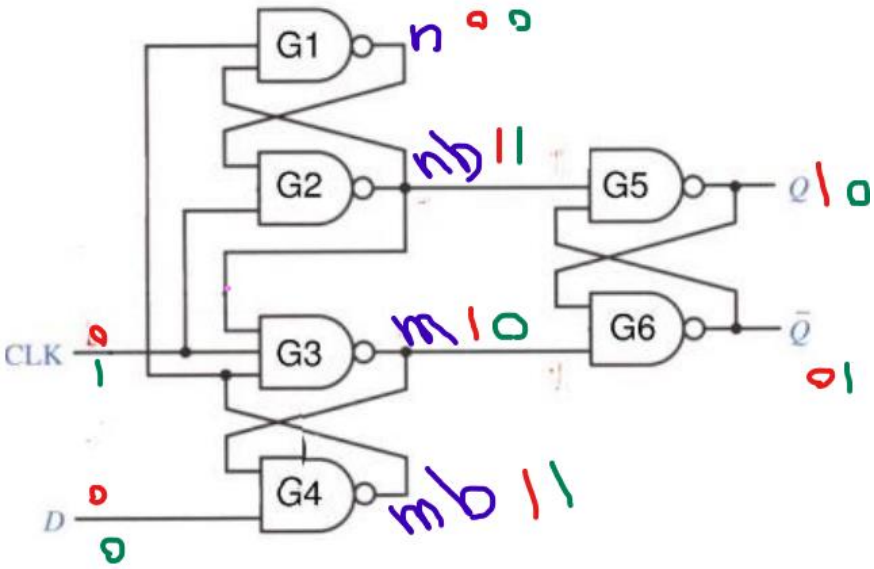
When there is posedge clock, data will be on output.

In 400ns, when clock changes to 1, the amount of data will be on output which 1 but with delay of 24ns because after changing clock G2 and G3 will be active. G2 is related to q. after 12nn it will calculate nb then G5 will be active and it takes more 12ns for calculating q. after changing q, one more 12ns is needed for calculating qb which is G6's duty.

While clock is still 1, data is changed to 0 in 600ns but based on wave forms, q and qb won't change. Because it's the way that an edge trigger d flip flop works. Outputs that are q and qb will change on posedge of clock but here there's no posedge.

On 800ns clock will change to 0 and nothing happens for outputs. While clock is 0 data changes to 1 and 0 but still nothing happens because there isn't any posedge of clock yet.

On 1400ns clock changes to 1 so there is a posedge of clock and based on table data will go on output which is 0 but here qb will change after 24ns and then on 36ns q will change but why?



Red: previous values

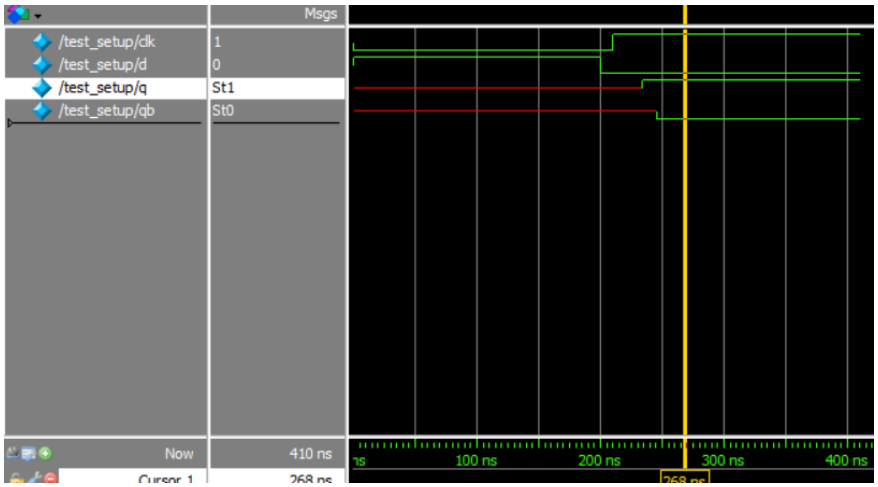
Green: new values

Based on the diagram, with changing clock to 1, G3 and G2 will be active but output of G2 that is nb won't change while after 12ns output of G3 which is m will change. Then G6 will be active. So, after 12ns qb will be set to 1. Now G5 will be active so after one more 12ns q will become 0.

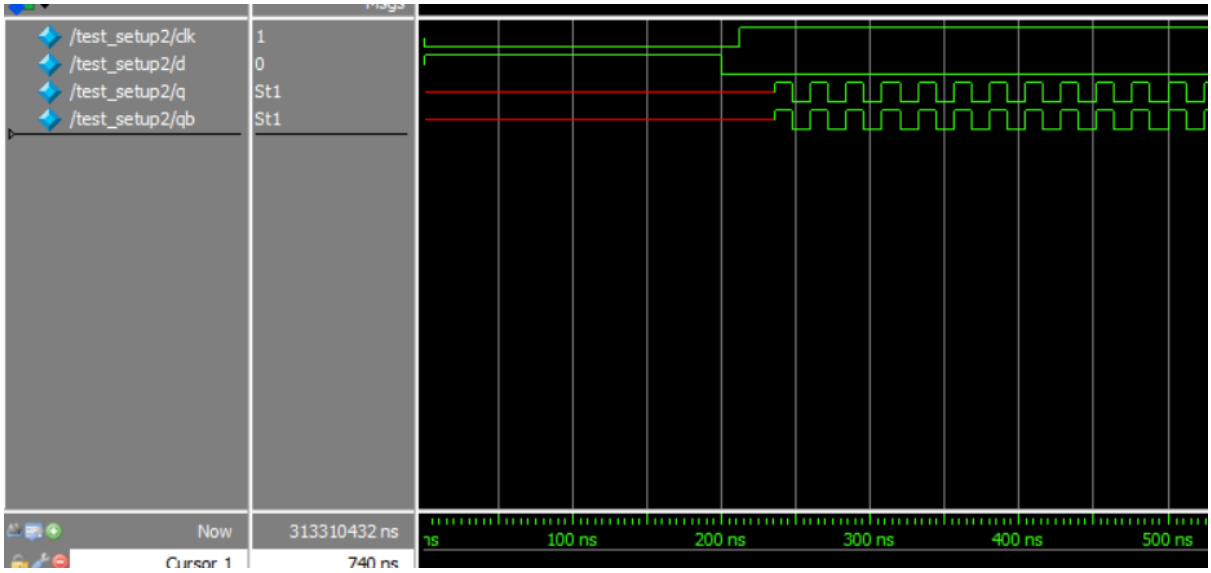
Part c)

As we know, gates have delay. When d changes, G4 needs 12ns for calculating mb. If posedge of clock happens while G4 is calculating mb, based on the time, there will be some unwanted situations.

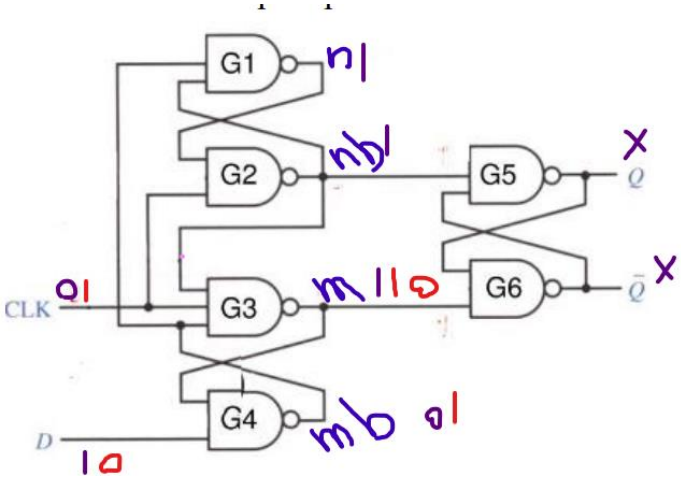
Set up time is 12ns. Based on modelsim, if posedge of clock happens less than 12ns after changing d, previous value of d will go on output instead of new one. But if posedge of clock happens exactly 12ns after changing d, bad things happen.



Here, 10ns after changing d, there is posedge of clock, which based on wave form, previous value of d which is 1 is on the output q.



In this case, posedge of clock comes exactly 12ns after changing d, which the result is bad as q and qb are toggling and are equal.



Purple: before posedge of clock

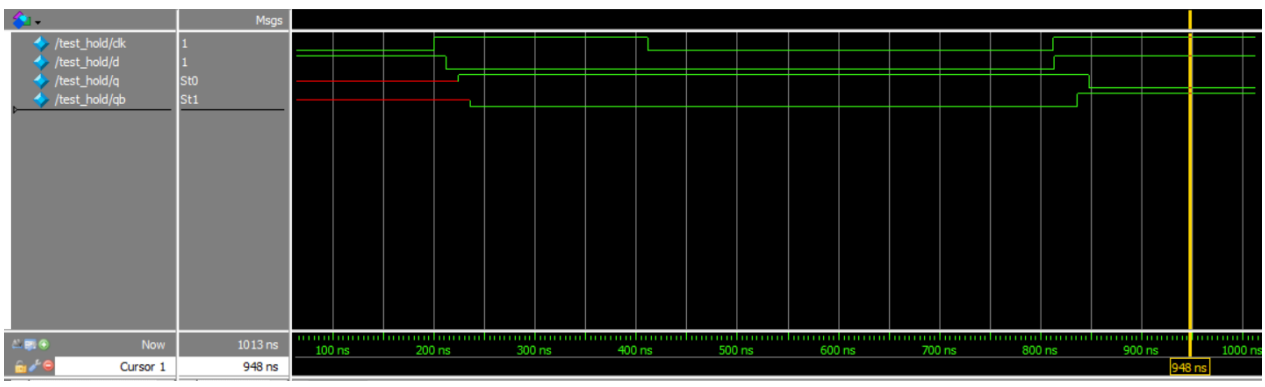
Red: after posedge of clock

Based on the diagram when clock changes to 1, mb was 0 so m will be 1, but simultaneously G4 calculated mb that changes to 1 so again m changes. So, this will cause toggling for outputs

Part d)

Based on the diagram and gate level, previous data should stay 12ns. Because we should wait till G3 calculate the output. But based on modelsim, it's ok to change the data whenever we want, because I think it will let the gate to calculate its output without paying attention to G4 which change the input of G3 while G3 is calculating.

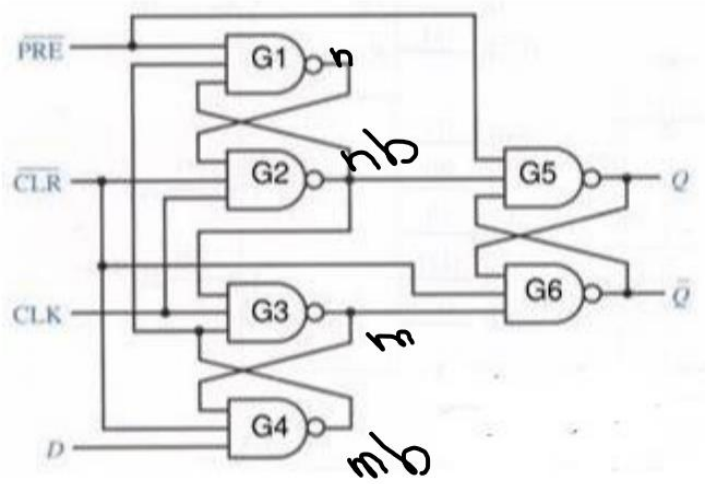
Wave form of testing 12ns and 1ns:



Based on the wave form, whenever d has changed, its previous value went on output.

Question 3:

Part a)

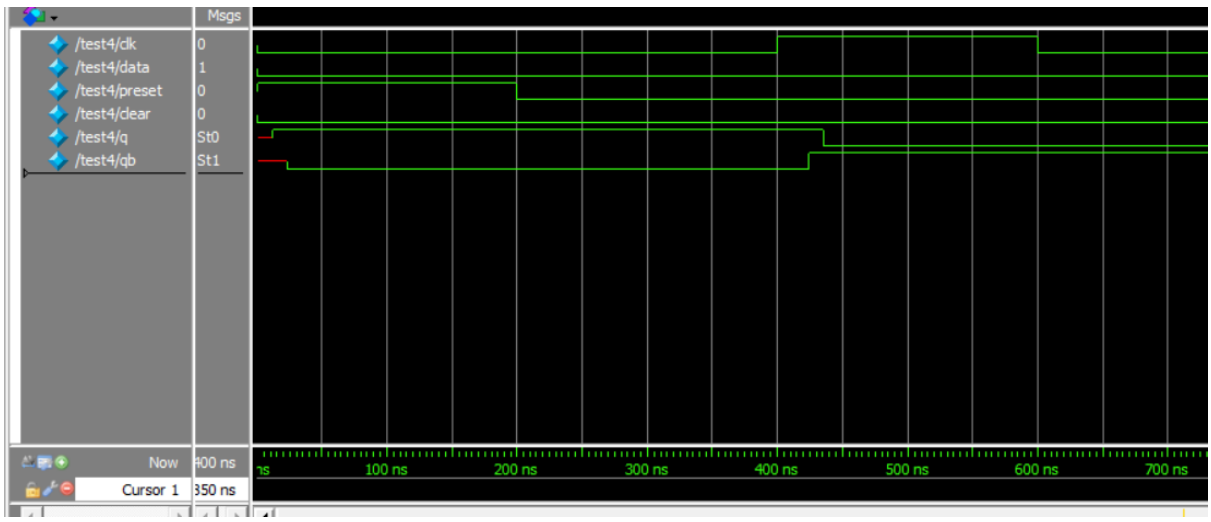


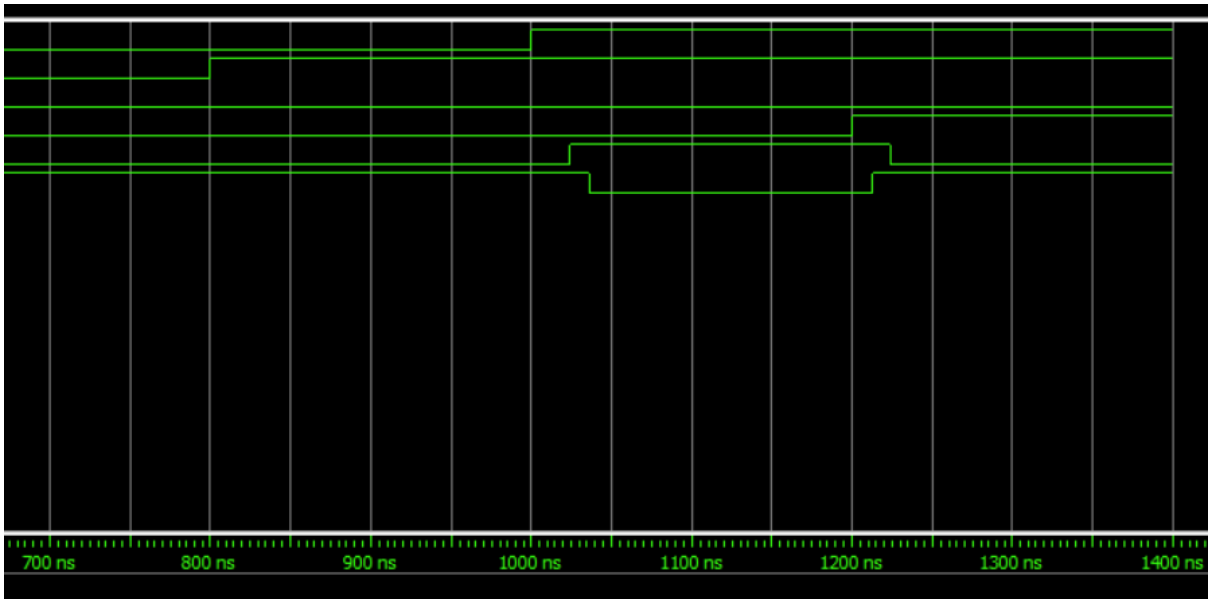
Based on this diagram, SR latches are connected to inputs and outputs. The table of this circuit is as below:

PRE	CLR	CLK	data	Q	~Q
1	0	X	X	1	0
0	1	X	X	0	1
0	0	0	X	Q	~Q
0	0	1	d	d	~d
1	1	X	X	invalid	

In this way it's possible to test the circuit. The situation that both preset and clear are active is invalid but we will test this.

Part b)





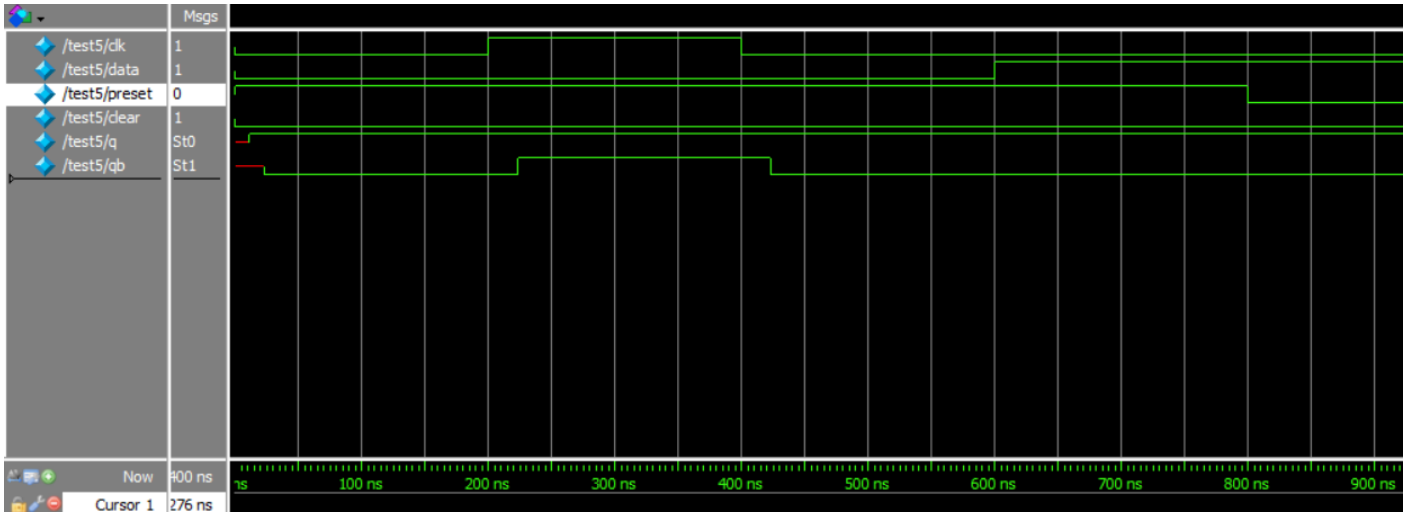
At first clock is zero but preset is 1 so the output will set to 1. As preset input is connected to G5 directly, after 12ns q will become 1. After that qb will change to 0 after one more 12ns which means 24ns from the start, as G6 is connected to q. then preset changes to 0 because we want to test the clocking of d input. Based on the table clear and preset priority is more than clock and data. So, when one of them is 1, it will choose the value of output with out paying attention to clocking and data.

On 400ns, there is posedge of clock so data which is 0 will go on the output. As G3 changes its output at first qb will change after 24ns, after qb, q will change in 36ns.

Then on 600ns clock will be 0 and then data will become 1 and in next posedge of clock which is on 1000ns, data will go on output.

Then in 1200ns clear input will be active which makes the output 0. As clear in connected to G6 directly, after 12ns qb will be 1 and 12ns after it which is totally 24ns q will be 0.

Part c)

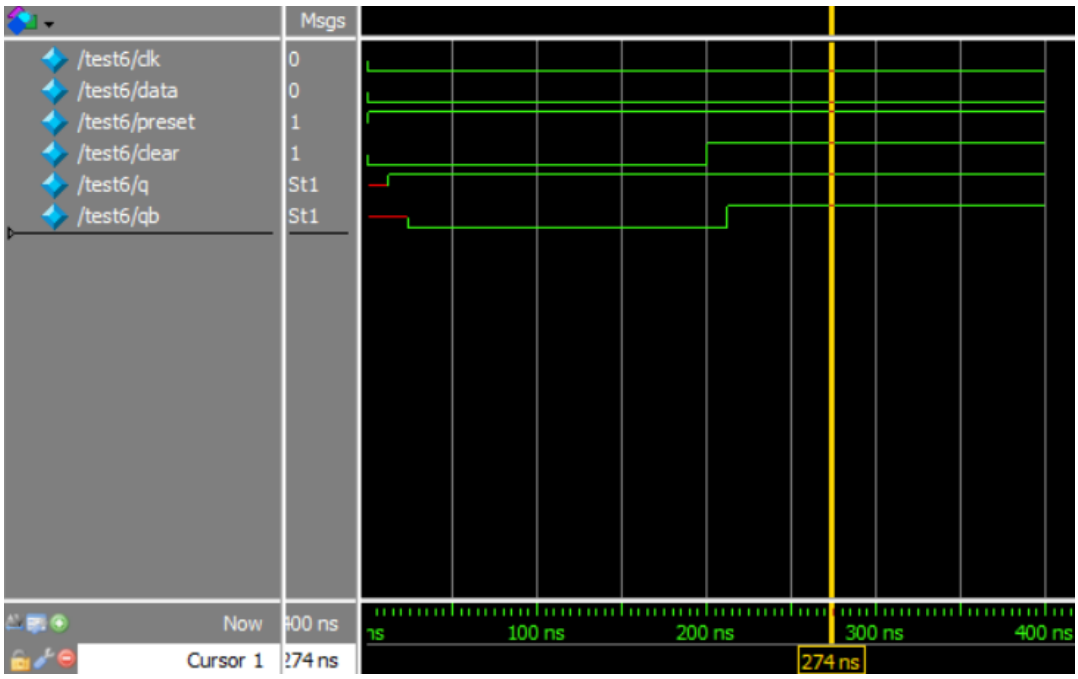


At the beginning, preset is 1 so based on the delay that was described, output changes to 1. While preset is active, there is posedge of clock. But based on wave form, q won't change to data, as preset is active. But qb changes to 1 as clock is related to it. Preset as no effect on qb as it's directly connected to G5 which is for q. so simultaneously q and qb are 1 which means memory loss. When clock changes to 0 qb will become 0 as well.

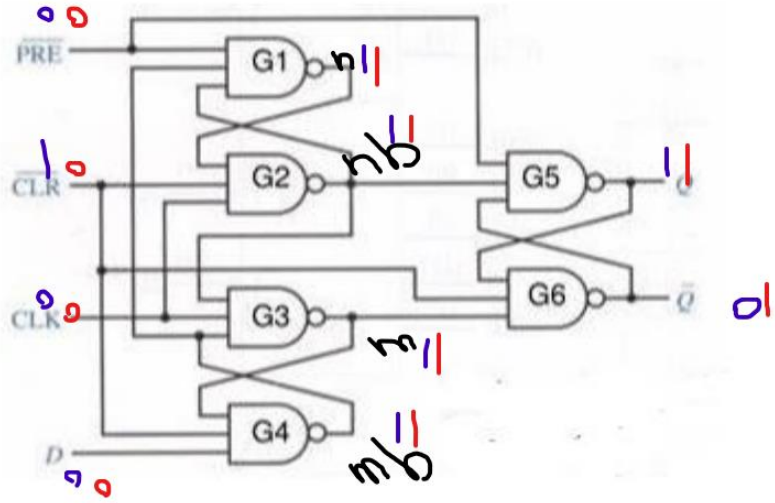
Then preset is inactive and clear is active so q will become 0. While clear is active, there is posedge of clock which doesn't change the output as clear is active.

So, preset and clear have higher priority as they are directly connected to G5 and G6 which calculate the values of q and qb.

Part c)



At the beginning, preset is active, so q is 1 and qb is 0. While preset is active, we make the clear active too. So, q and qb will become 1 simultaneously which is not true. But how these outputs will be calculated?



Blue: preset = 1 clear = 0

Red: preset = clear = 1

When preset is 1 as we connect ~pre to the circuit, while it's 1 q will be 1 too.

When we make clear active as well, just one of the G6 inputs will change. So, qb will become 1. But this situation has no effect on q as one of its inputs which is ~pre is still 0 so it will delete the effect of qb on q.