(*) Building an 8-bit Breadboard Computer

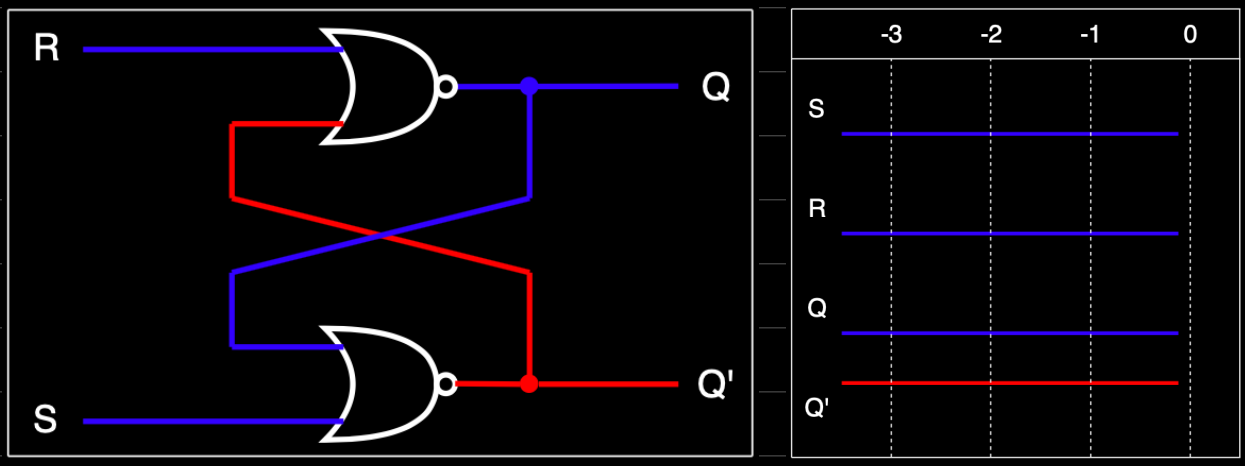
by Ben Eater

(https://youtube.com/playlist?list=PLowKtXNTBypGqImE405J2565dvjafglHU)

^ NOR-Gate SR Latch

(https://youtu.be/KM0DdEaY5sY)

Let the initial STABLE state be S = 0, R = 0, Q = 0 and Q' = 1 (CLEAR).



Let the PROPAGATION delays of the upper and lower NOR gates be 2 units and 3 units, respectively.

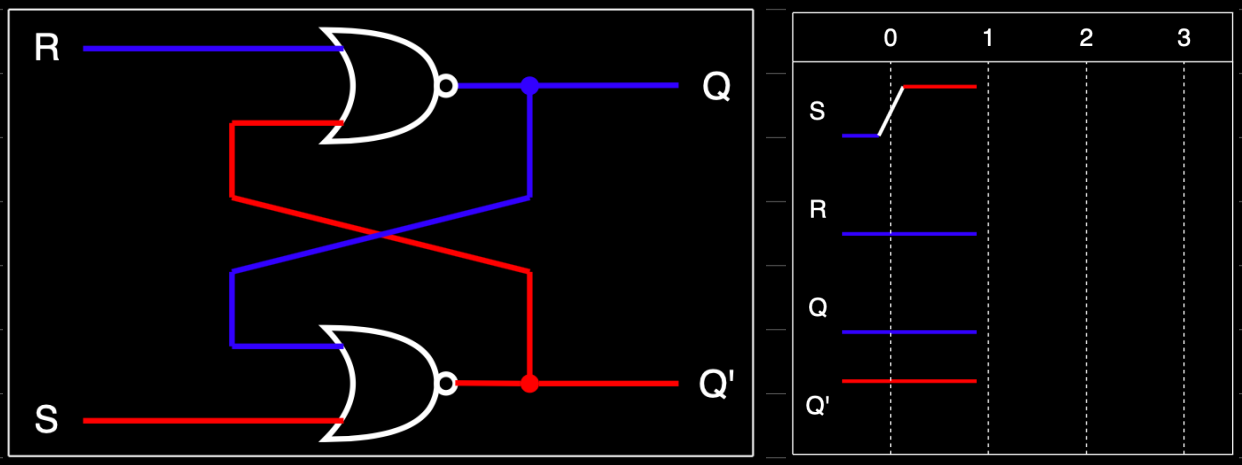
So, the EFFECTS of the changed inputs will be visible only AFTER the propagation delays.

In order to draw the TIMING diagrams, for every logic gate, look at its inputs at time (t - T), where T is its propagation delay.

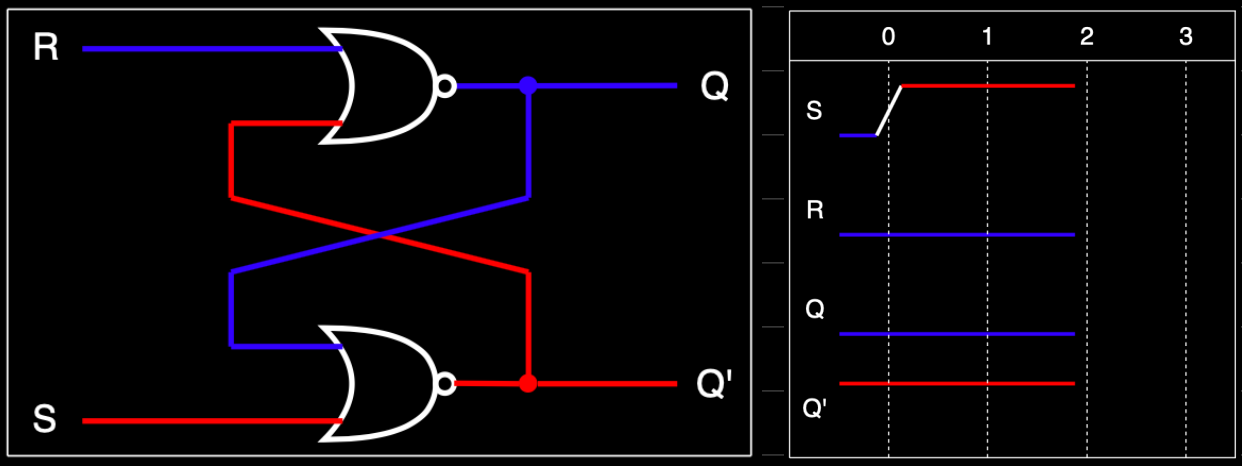
For eg., if the propagation delay of a logic gate is 3 units, then its OUTPUT at t = 3.5 units (say) depends upon its INPUTS at t = 0.5 units.

Now, let S get changed to 1 at t = 0 (in order for the latch to become SET).

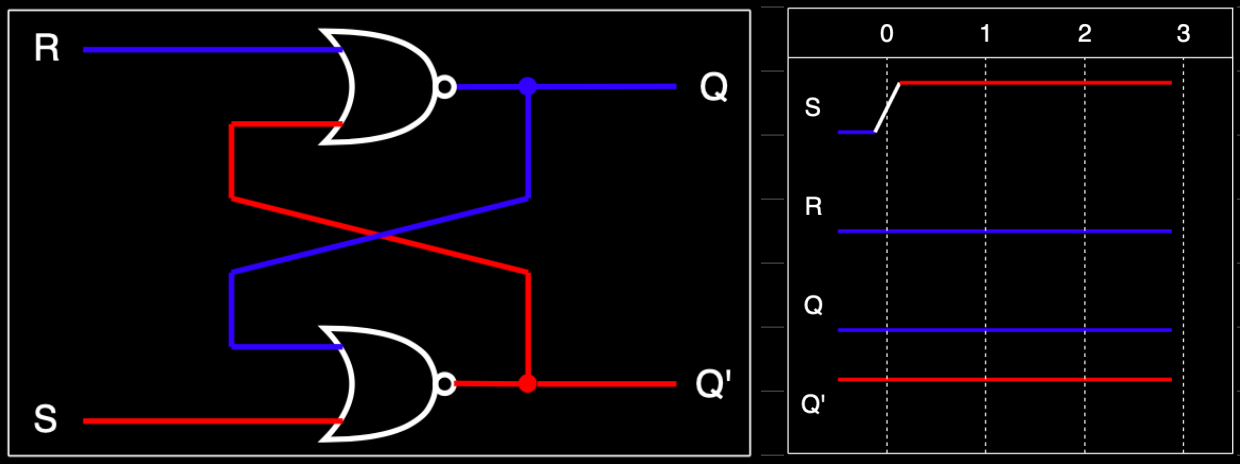
From t = 0 to t = 1 :-



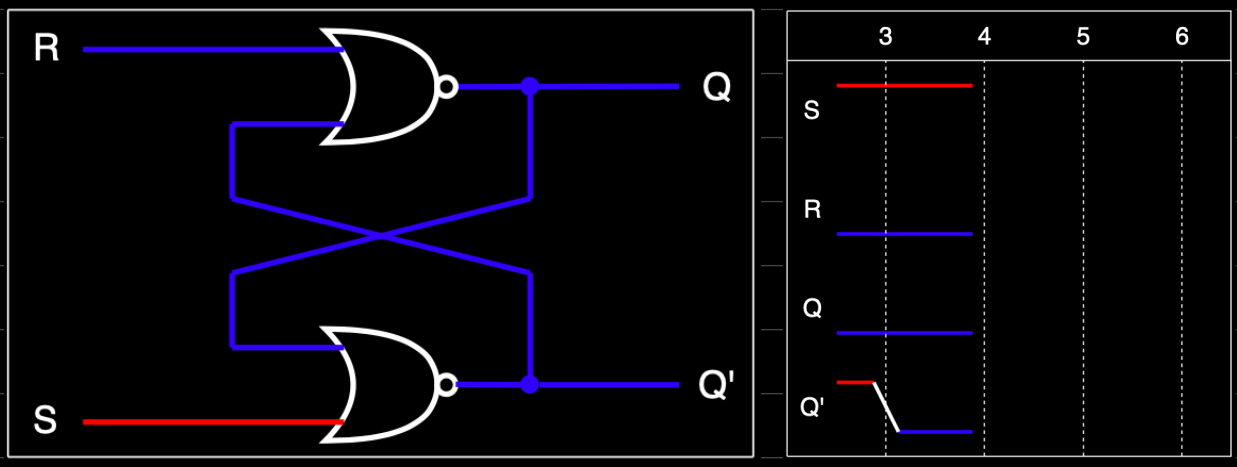
From t = 1 to t = 2 :-



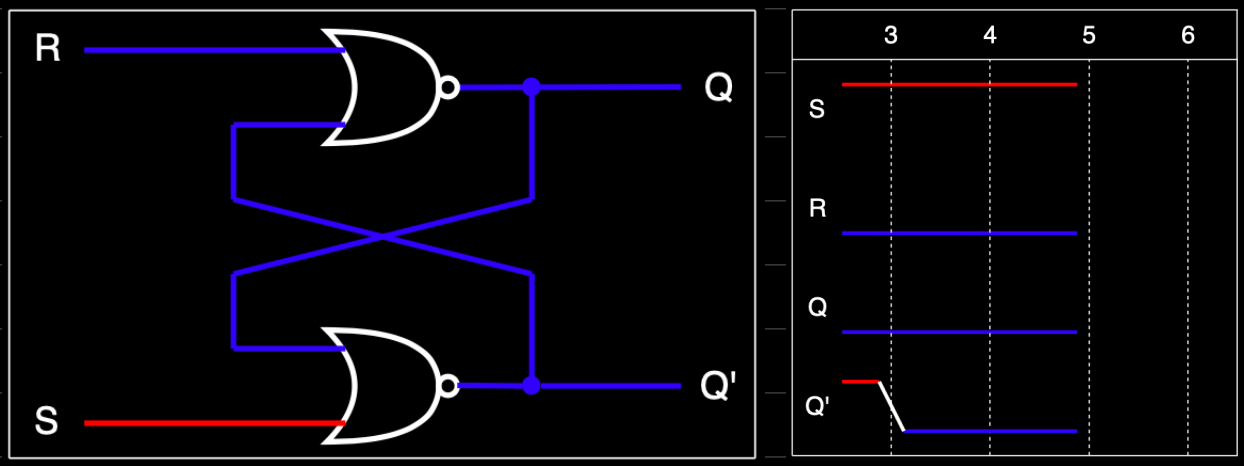
From t = 2 to t = 3 :-



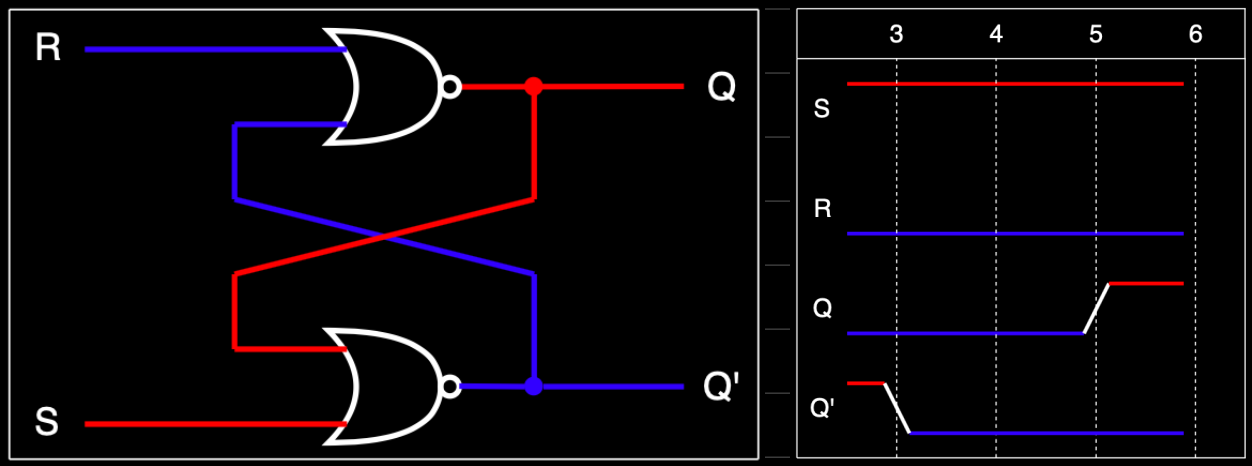
From t = 3 to t = 4 :-



From t = 4 to t = 5 :-



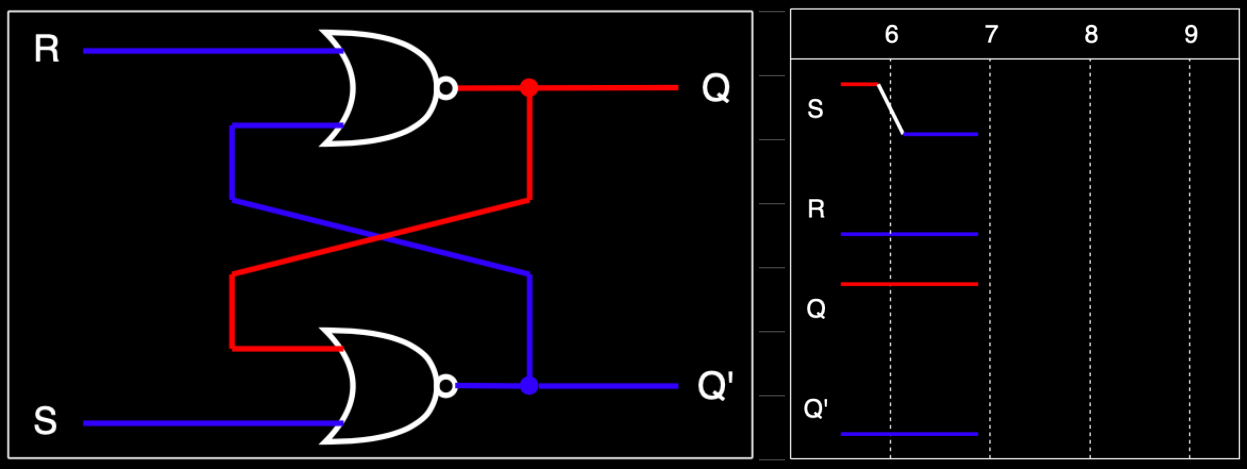
From t = 5 to t = 6 :-



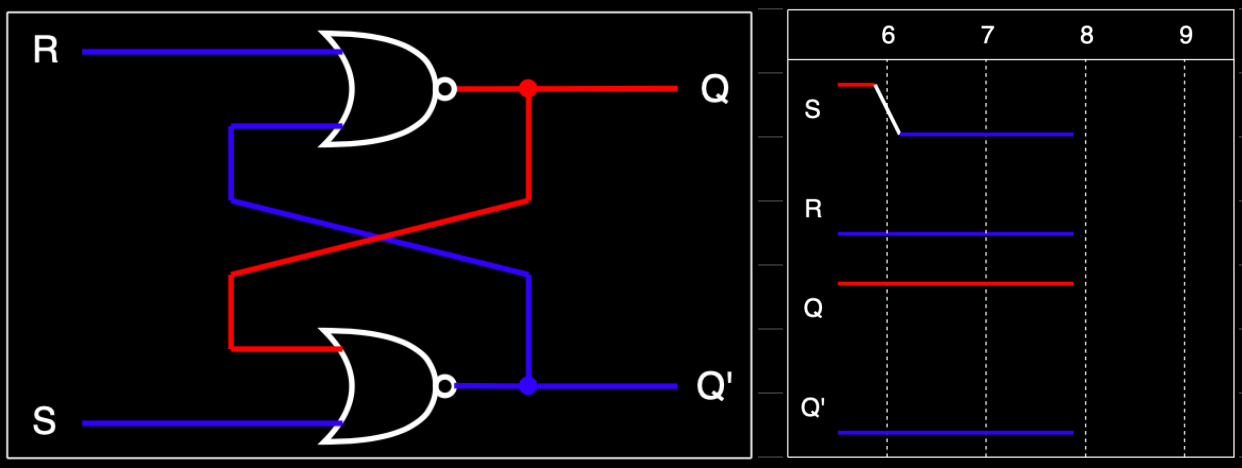
Now, the latch will stay STABLE in the SET state, even if S gets changed back to 0.

Let S get changed back to 0 at t = 6.

From t = 6 to t = 7 :-



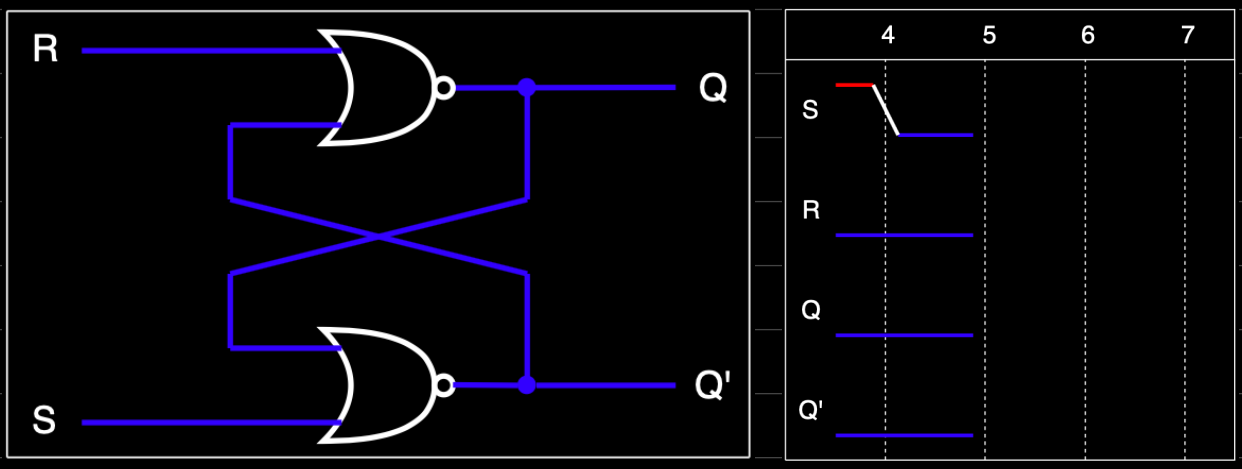
From t = 7 to t = 8 :-



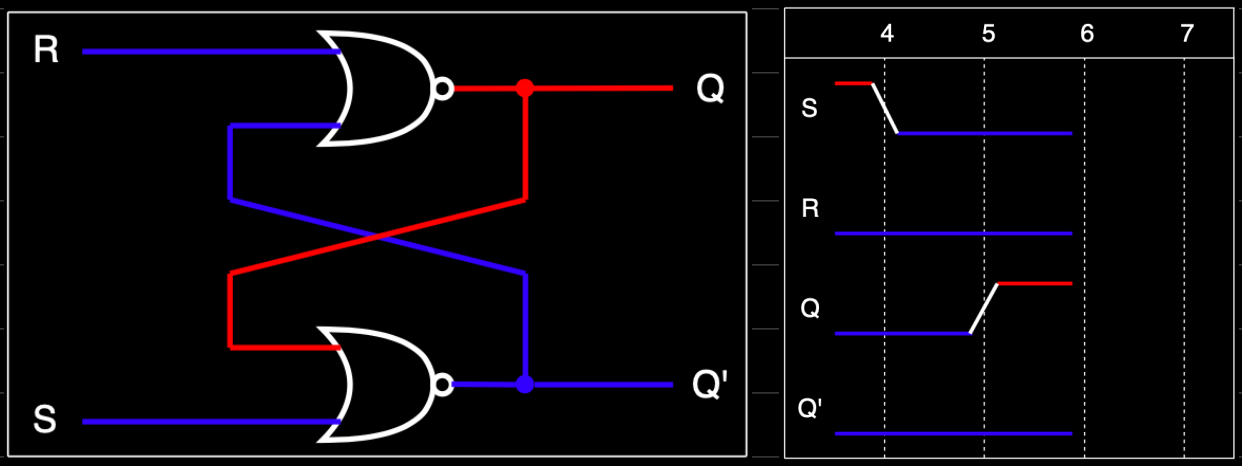
Hence, the latch will stay STABLE in the SET state, as long as S was maintained at 1 for a LONG ENOUGH time, i.e. if S gets changed back to 0 AFTER t = 5, then the latch will result in a STABLE SET state.

Now, let S get changed back to 0 at t = 4, instead of at t = 6.

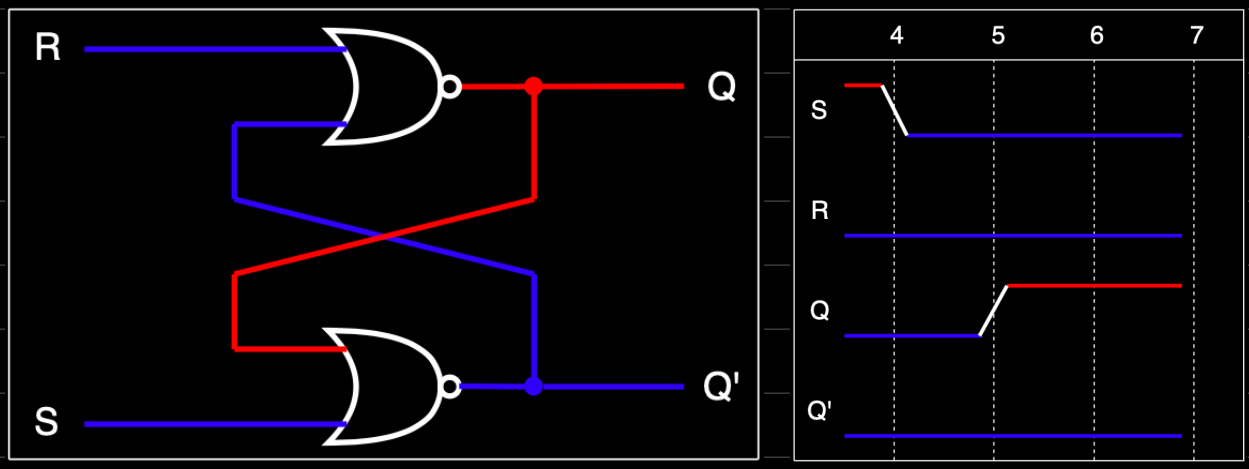
From t = 4 to t = 5 :-



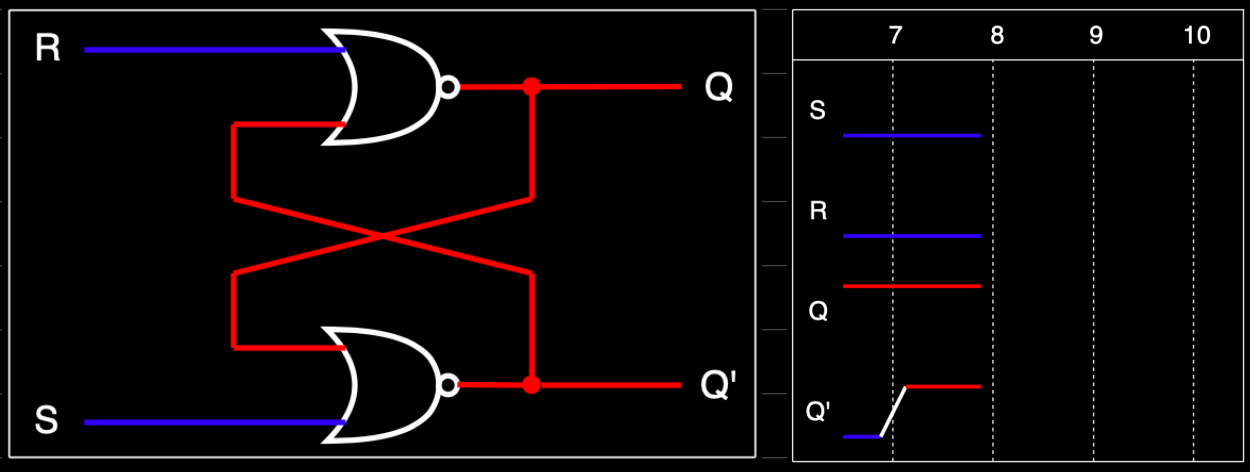
From t = 5 to t = 6 :-



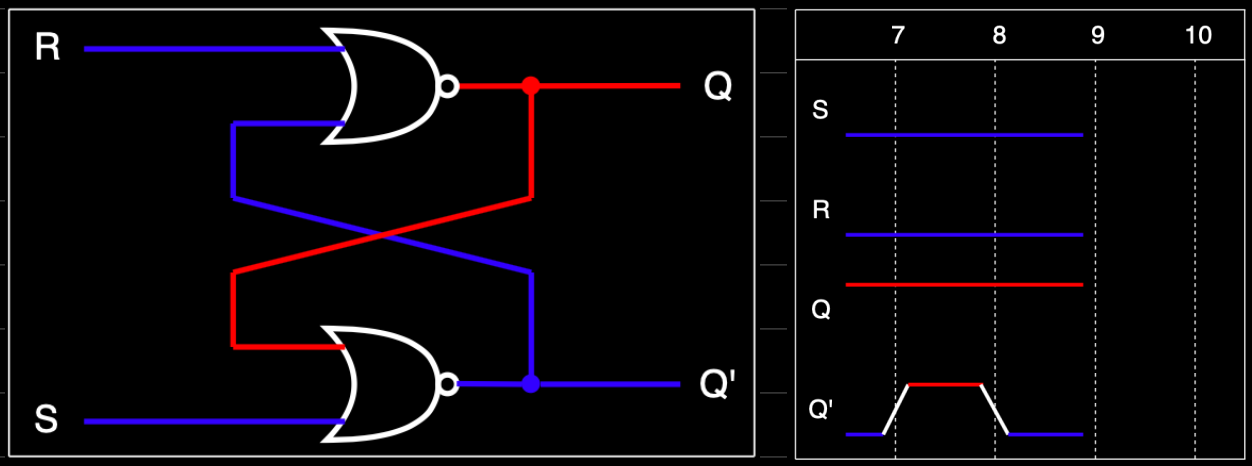
From t = 6 to t = 7 :-



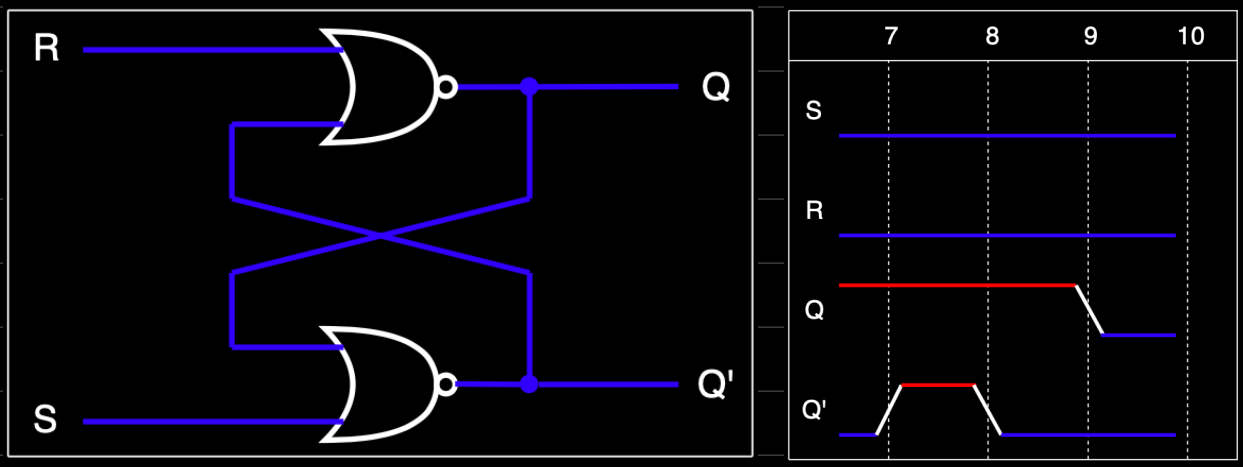
From t = 7 to t = 8 :-



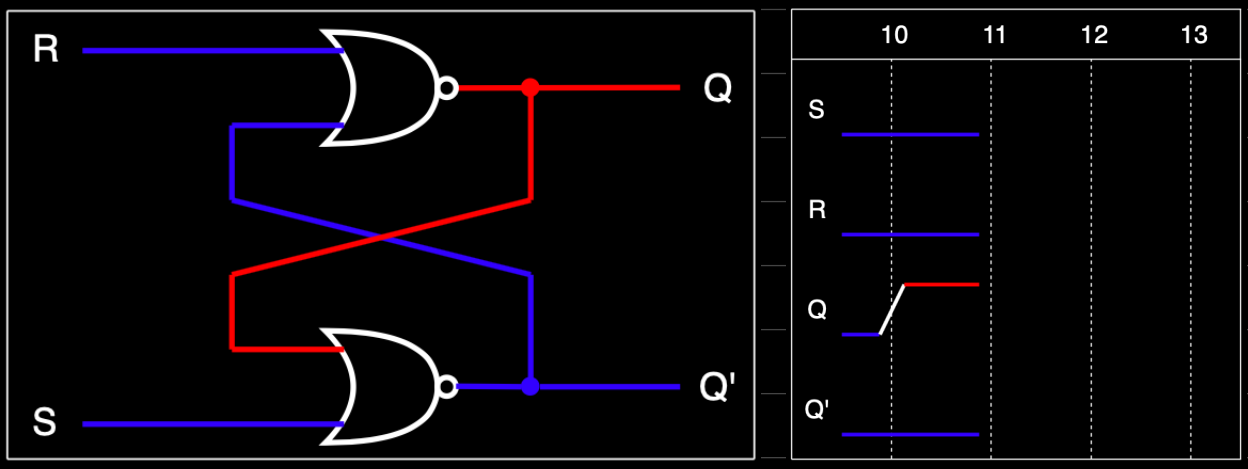
From t = 8 to t = 9 :-



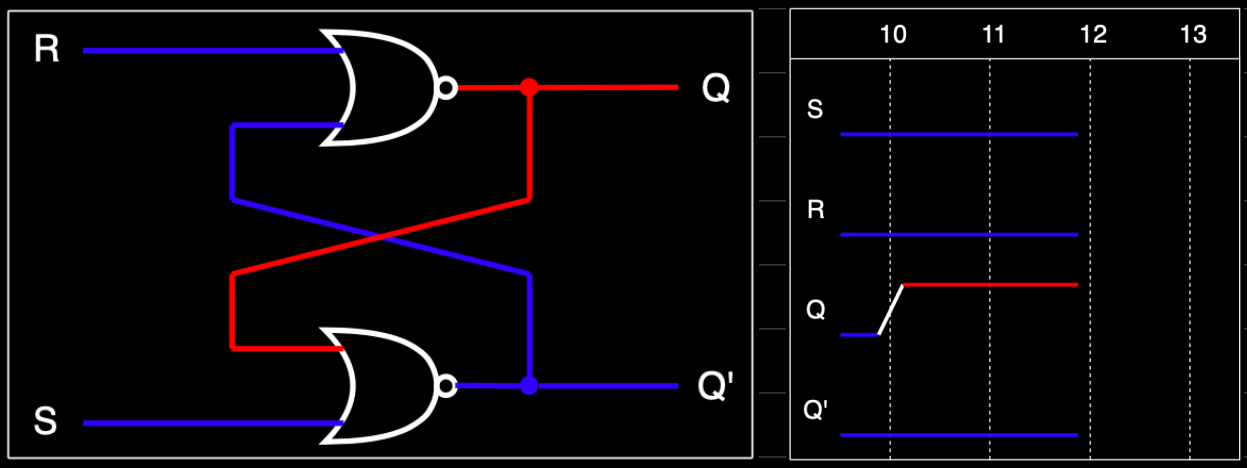
From t = 9 to t = 10 :-



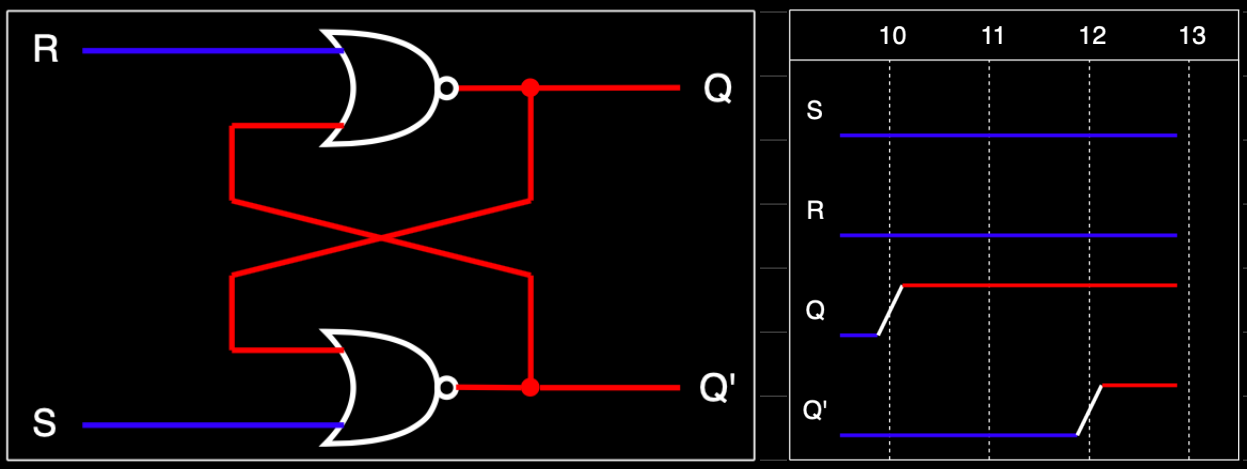
From t = 10 to t = 11 :-



From t = 11 to t = 12 :-



From t = 12 to t = 13 :-

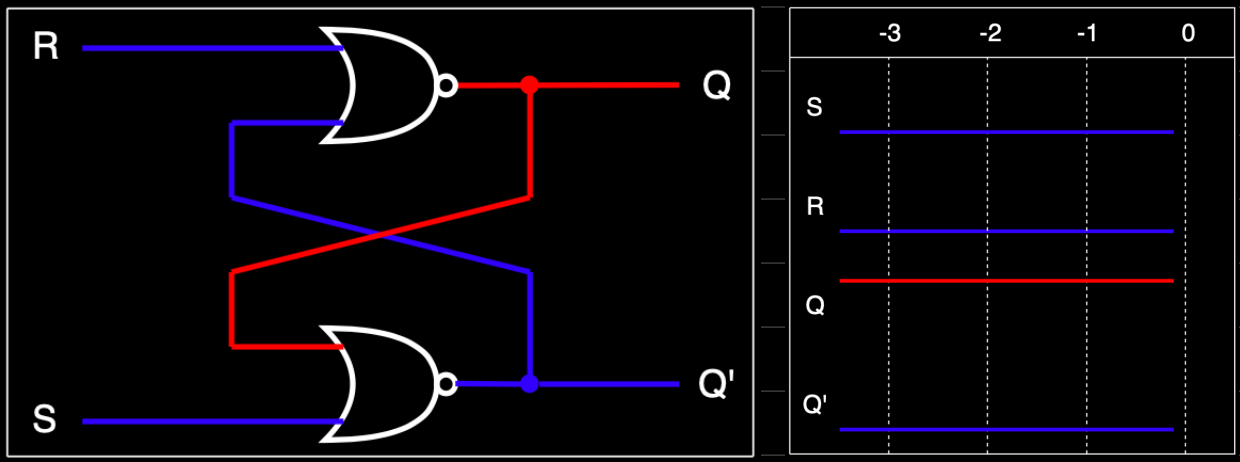


The latch will keep oscillating like this FOREVER. Previously, when the latch was being SET, it temporarily switched to an INVALID state, i.e. Q = Q' = 0, but then it became STABLE in the SET state. But, in this case, the latch will NEVER become STABLE in ANY state. Therefore, if S is NOT maintained at 1 for a LONG ENOUGH time, then problems like this may occur.

It should be noted that the above analysis corresponds to an IDEAL version of logic gates. In real-world circuits, for EDGE cases like this, logic gates may NOT work as expected. For eg., even though ideally the latch should keep oscillating FOREVER, in real-world circuits, the latch MAY or MAY NOT keep oscillating due to the time it takes to turn on and off transistors, to charge internal capacitors, etc.
In real-world circuits, a change in the inputs for a SMALL enough time does NOT contain enough ENERGY to cause a logic gate to switch its output.

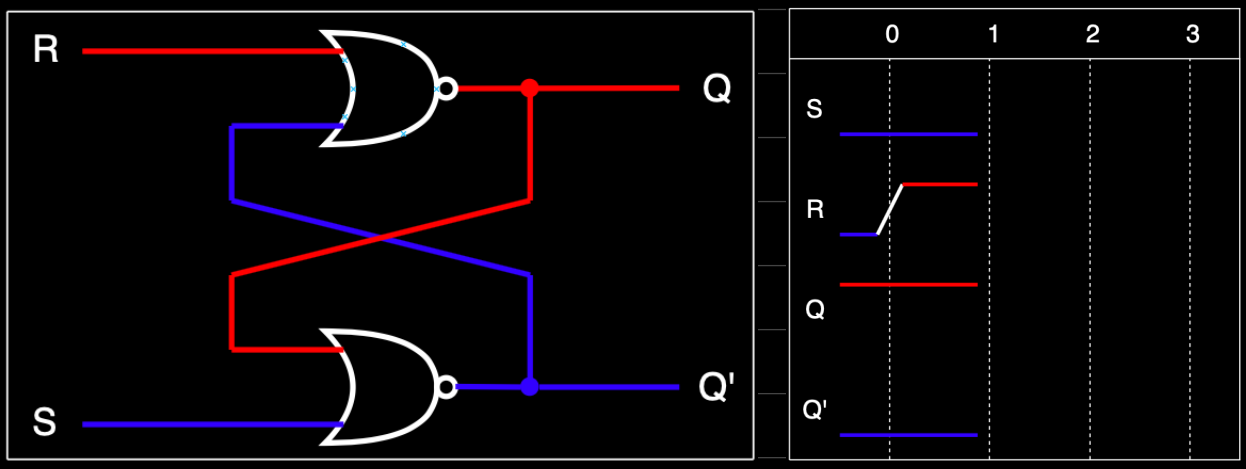
In any case, if used CORRECTLY, i.e. by keeping S/R high for a LONG ENOUGH time when setting/clearing the latch and by NOT keeping S & R high at the same time, then the latch will work AS EXPECTED in ideal as well as real-world circuits.

Now, Let the initial STABLE state be S = 0, R = 0, Q = 1 and Q' = 0 (SET).

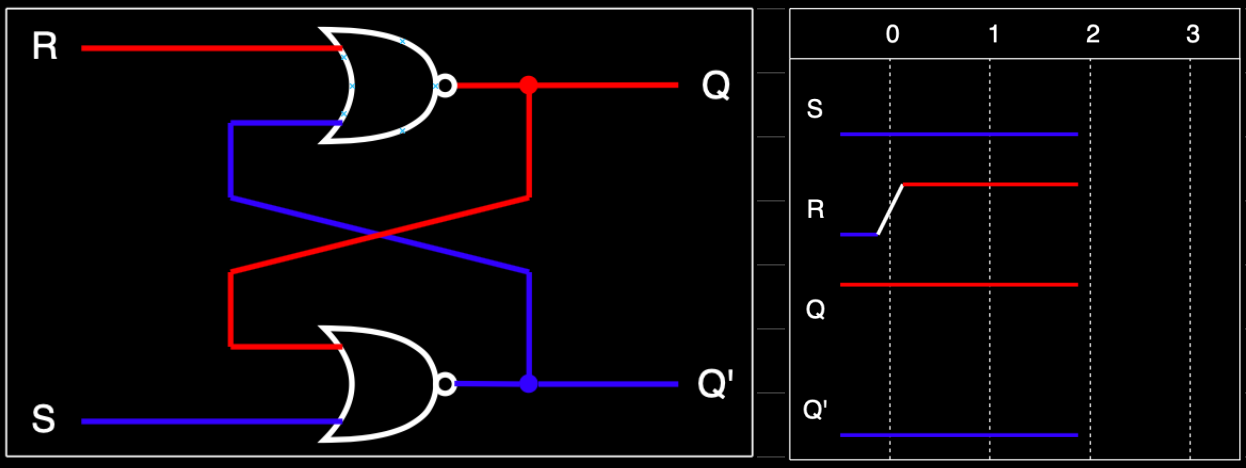


Let R get changed to 1 at t = 0 (in order for the latch to become CLEAR).

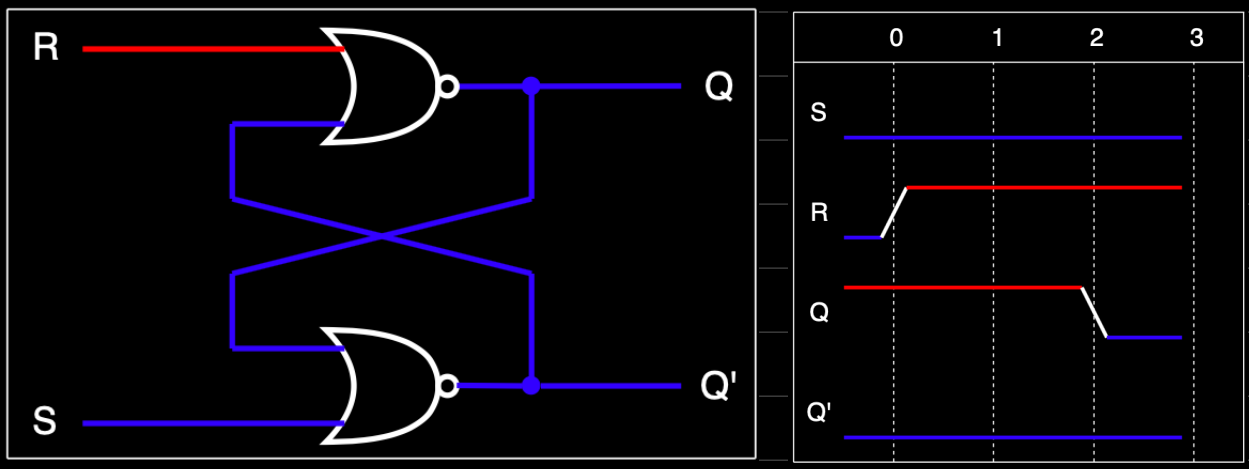
From t = 0 to t = 1 :-



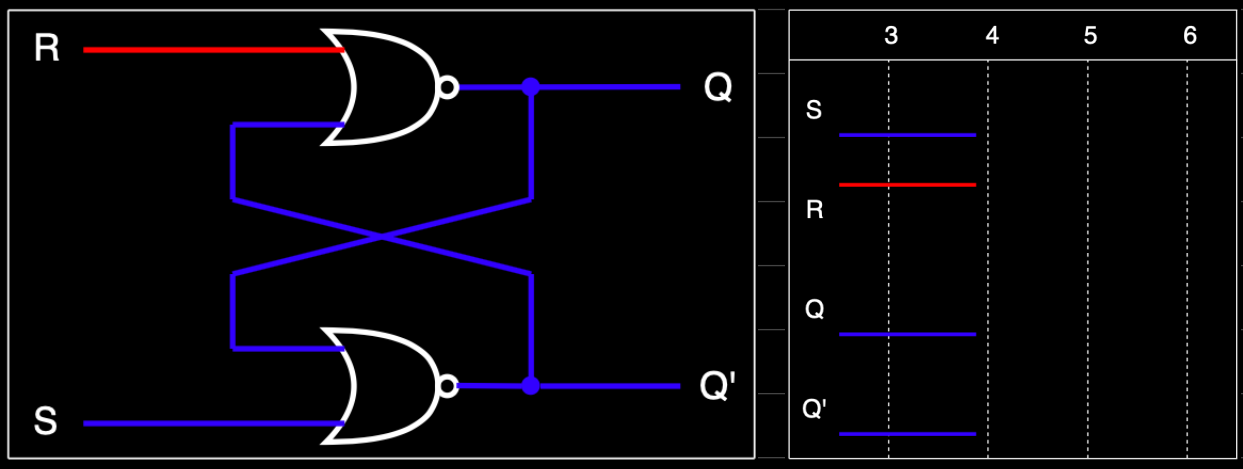
From t = 1 to t = 2 :-



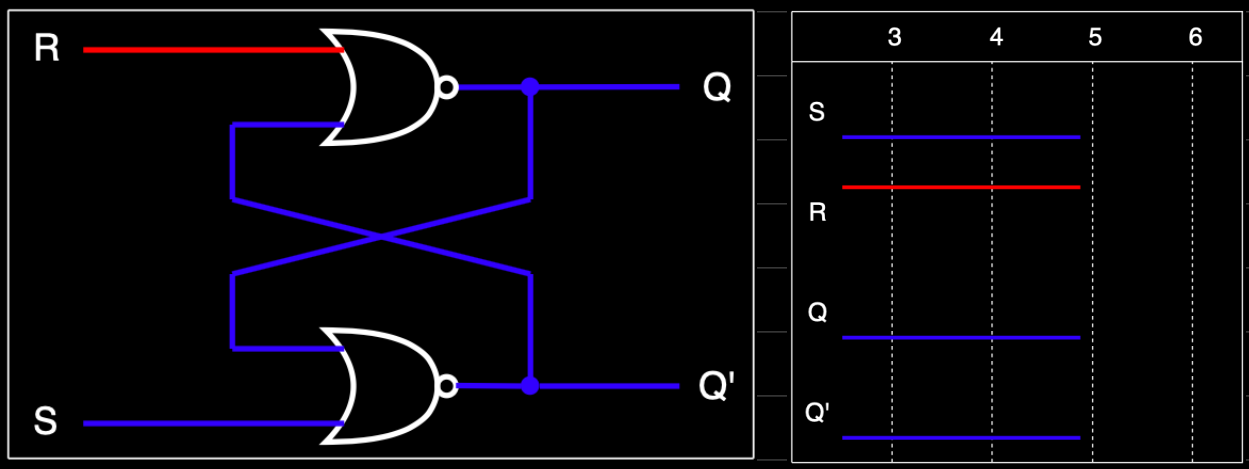
From t = 2 to t = 3 :-



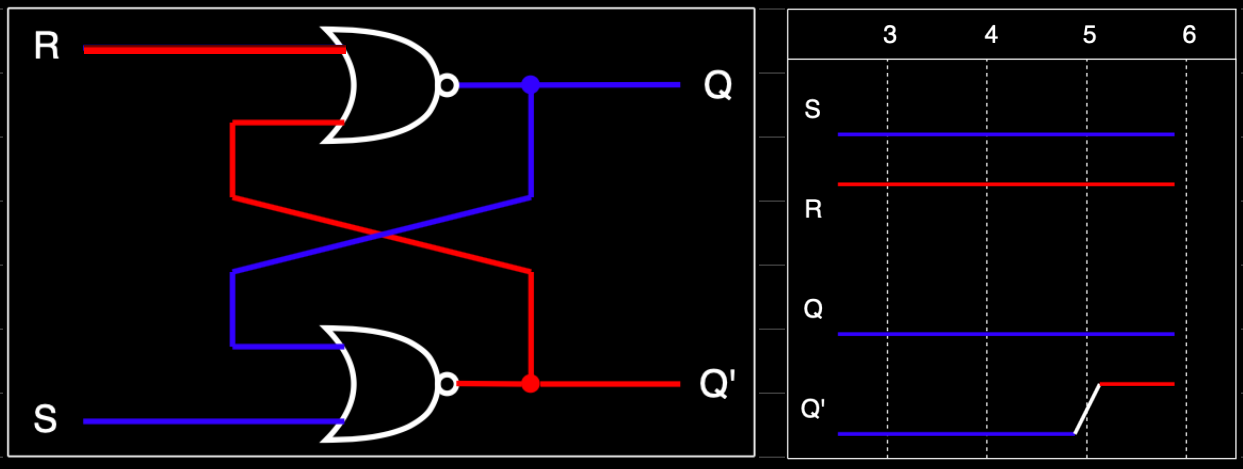
From t = 3 to t = 4 :-



From t = 4 to t = 5 :-



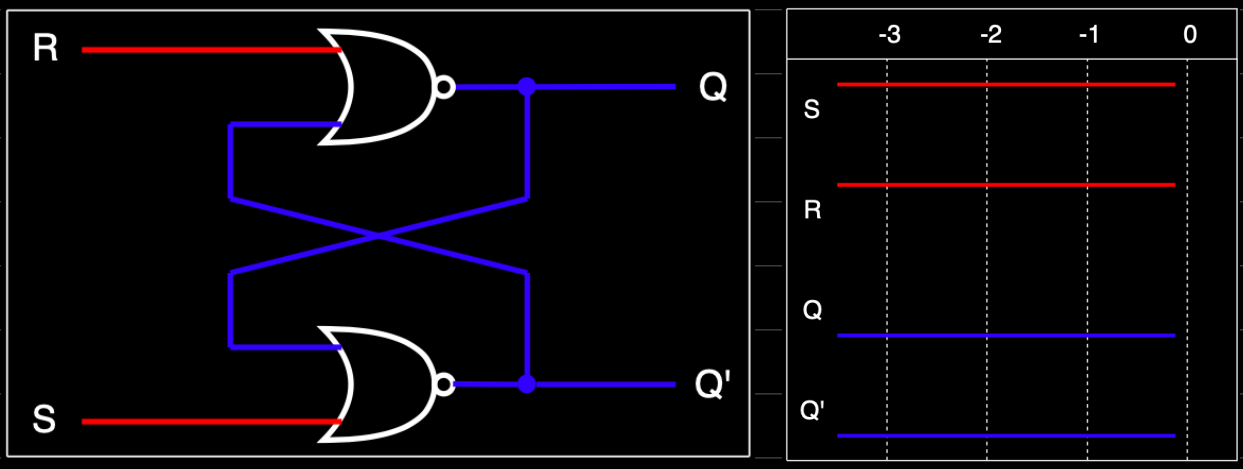
From t = 5 to t = 6 :-



Now, the latch will stay STABLE in the CLEAR state, even if R gets changed back to 0, as long as R was maintained at 1 for a LONG ENOUGH time, i.e. if R gets changed back to 0 AFTER t = 5, then the latch will result in a STABLE CLEAR state.

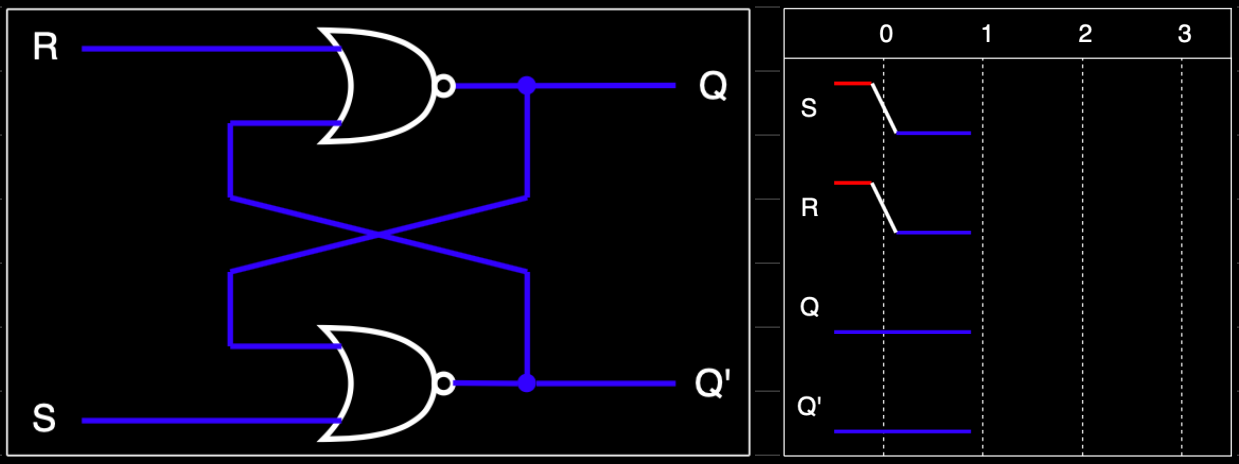
Therefore, if R is NOT maintained at 1 for a LONG ENOUGH time, then similar problems may occur like if S is NOT maintained at 1 for a LONG ENOUGH time when SETTING the latch.

Now, Let the initial STABLE state be S = 1, R = 1, Q = 0 and Q' = 0 (INVALID).

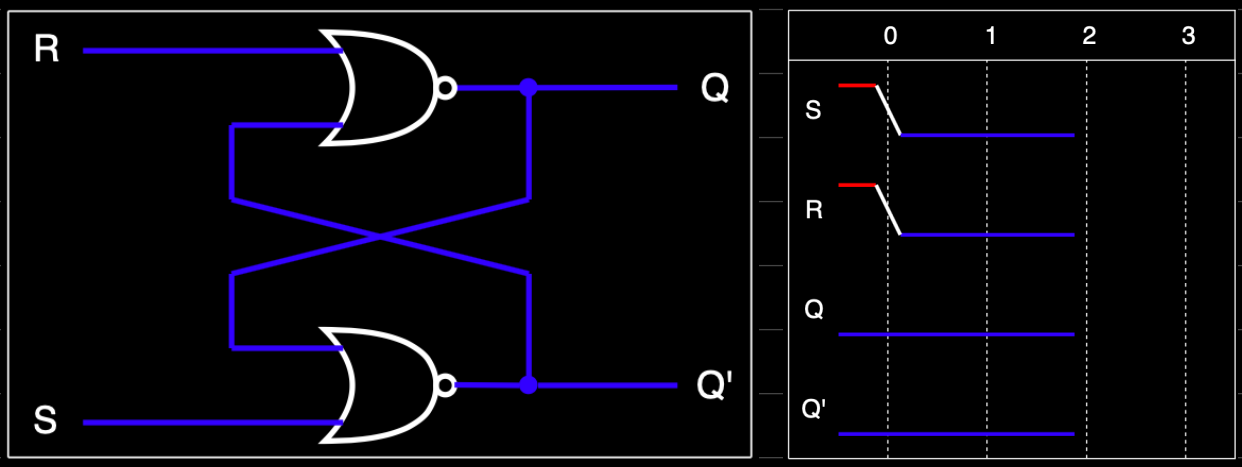


Let S & R both get changed to 0 at t = 0.

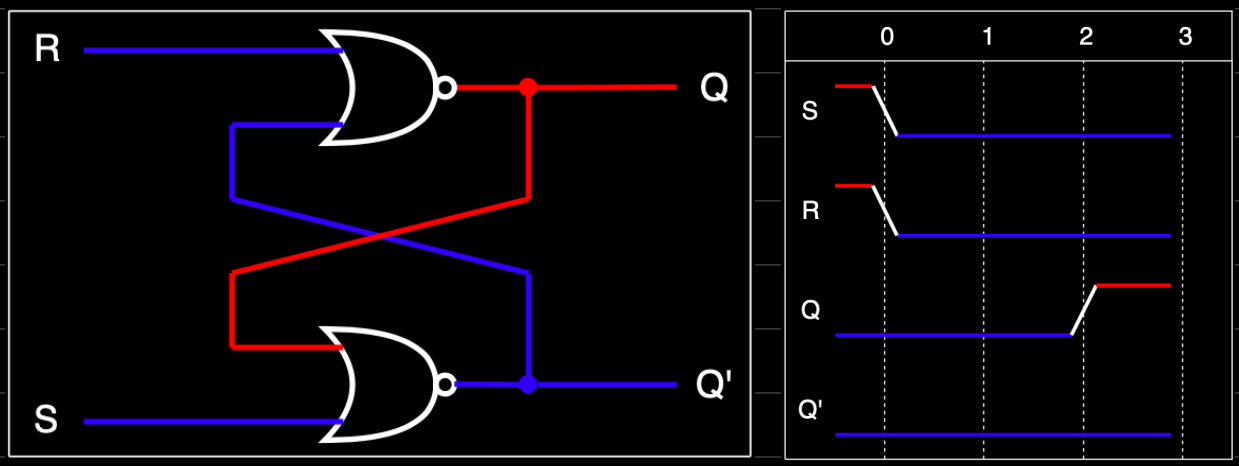
From t = 0 to t = 1 :-



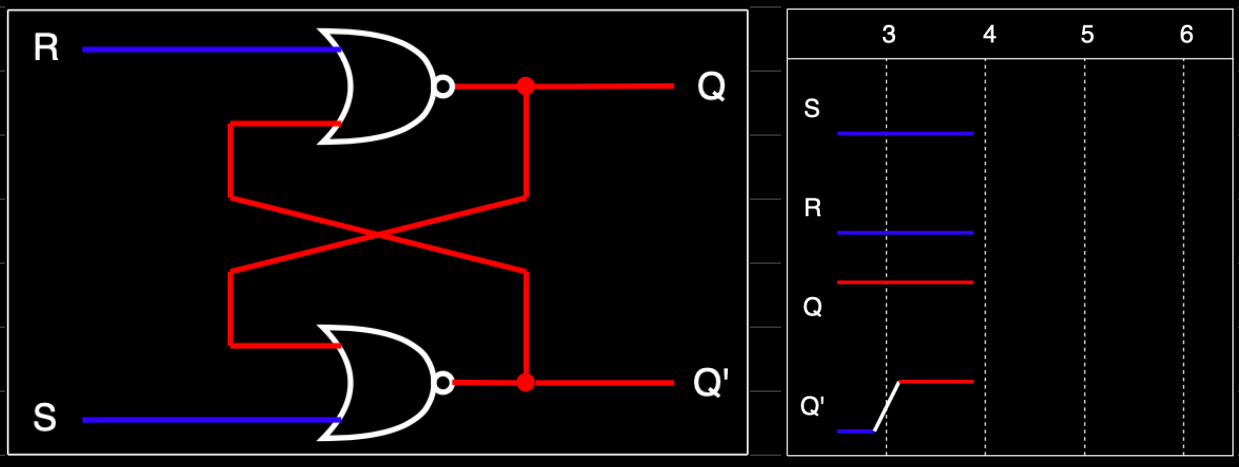
From t = 1 to t = 2 :-



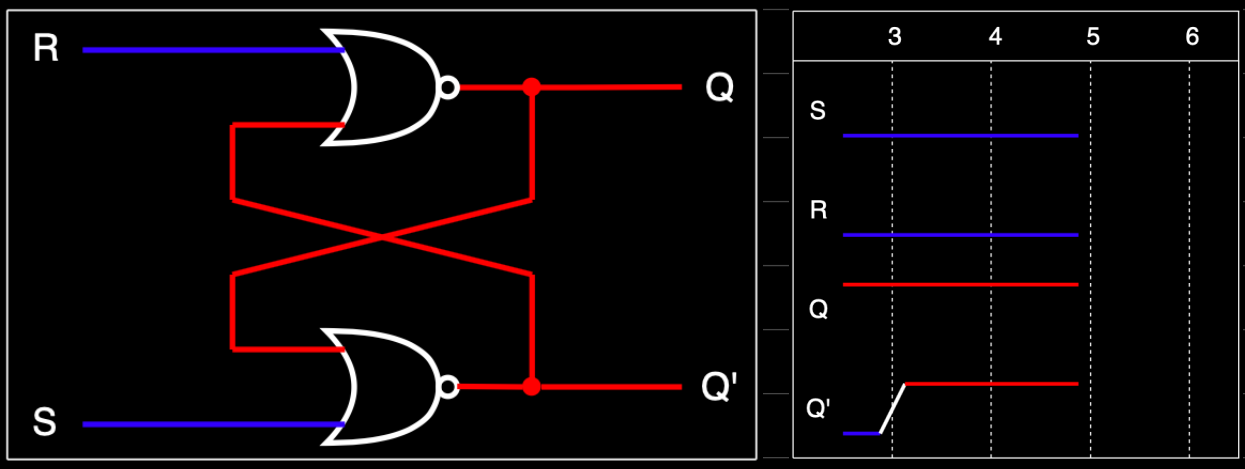
From t = 2 to t = 3 :-



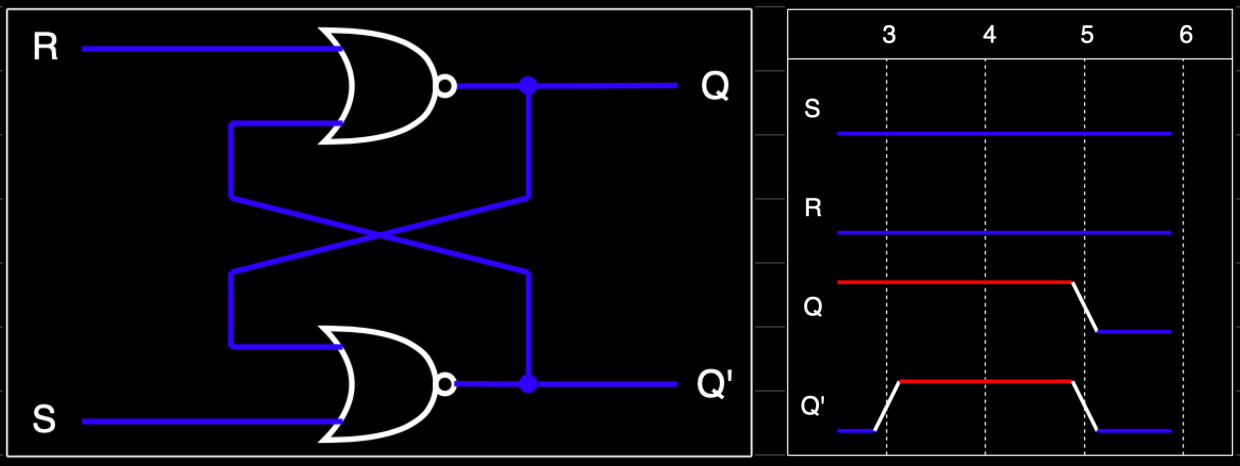
From t = 3 to t = 4 :-



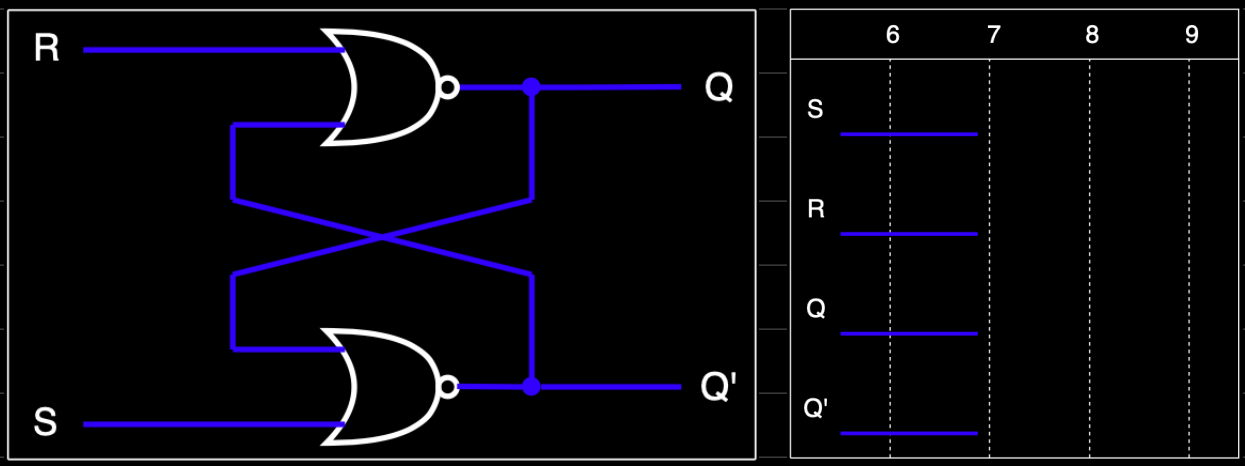
From t = 4 to t = 5 :-



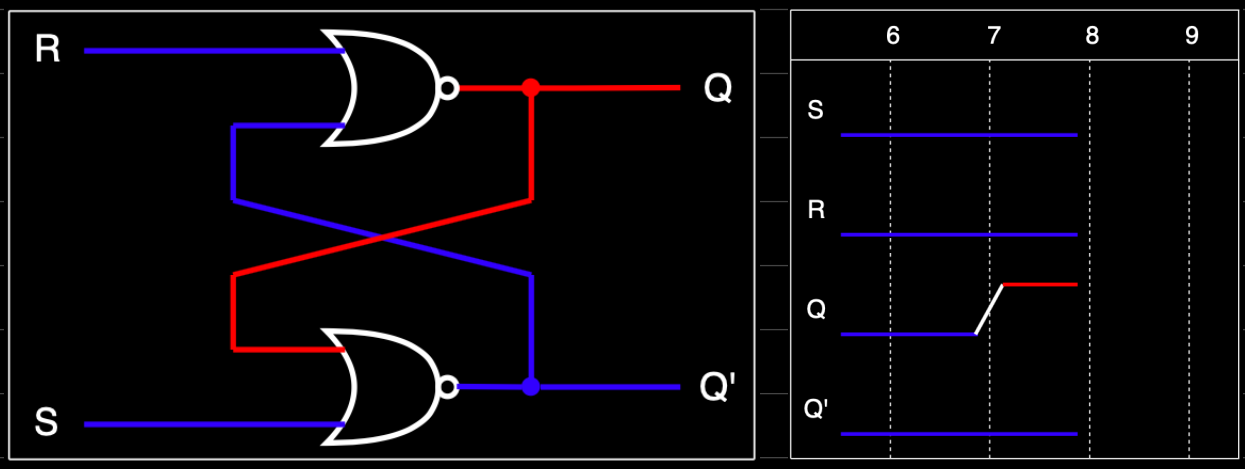
From t = 5 to t = 6 :-



From t = 6 to t = 7 :-



From t = 7 to t = 8 :-



The latch will keep oscillating like this FOREVER.

Therefore, if S and R both get changed to 0 after the latch is STABLE in the INVALID state of S = 1, R = 1, Q = 0 and Q' = 0, then problems like this may occur.

As stated previously, in real-world circuits, the latch MAY or MAY NOT keep oscillating.
In any case, the state of S = 1, R = 1, Q = 0 and Q' = 0 should be AVOIDED in ideal as well as real-world circuits.

It should be noted that if the initial STABLE state is S = 0, R = 0, Q = 1 and Q' = 0 (SET), and if S gets changed to 1, then the latch will REMAIN in the SET state.

Similarly, if the initial STABLE state is S = 0, R = 0, Q = 0 and Q' = 1 (CLEAR), and if R gets changed to 1, then the latch will REMAIN in the CLEAR state.

It should be noted that, for eg., while setting/clearing the latch, S/R was PURPOSEFULLY not changed back to 0 EXACTLY at t = 5 because doing so may work IDEALLY, but MAY or MAY NOT work in REAL-WORLD circuits.

To be answered later (?) :-

After a stable state of S = 0, R = 0, Q = 1/0 and Q' = 0/1, if S & R both get changed to 1 and then back to 0 without waiting for a long enough time, then what happens?

After a stable state of S = 1/0, R = 0/1, Q = 1/0 and Q' = 0/1, if R/S gets changed to 1 and then after waiting for a long enough time or without waiting for a long enough time, S/R gets changed to 0 and stays at 0 for a long enough time, then what happens?
- The latch goes into a stable clear/set state.
This is because after S and R become constant at 1/0 and 0/1, respectively, then no matter the current state of the latch, stable, oscillating or even floating, the latch will result in a stable set/clear state after a long enough time.
For eg., if one input of a NOR gate is 1, then after its propagation delay, its output will become 0, irrespective of the other input (even if that other input is floating).

After a stable state of S = 1, R = 1, Q = 0 and Q' = 0, if only S/R gets changed to 0, then what happens? - The latch goes into a stable clear/set state.
But, if the other input gets changed to 0 as well without waiting for a long enough time for the latch to go into a stable clear/set state, then what happens?

Etc.

^ NOR-Gate Gated D Latch

(https://youtu.be/peCh_859q7Q?si=M2RUMPAdl8BvUnTm)

To be answered later (?) :-

When the enable/clock is high and the latch is stable, then after changing D and keeping it constant for a long enough time, why does the NOT gate's propagation delay not cause problems?

The answer to this question is similar to the answer to the following question for the NOR-gate SR latch -

After a stable state of S = 1/0, R = 0/1, Q = 1/0 and Q' = 0/1, if R/S gets changed to 1 and then after waiting for a long enough time or without waiting for a long enough time, S/R gets changed to 0 and stays at 0 for a long enough time, then what happens?

- The latch goes into a stable clear/set state.

This is because after S and R become constant at 1/0 and 0/1, respectively, then no matter the current state of the latch, stable, oscillating or even floating, the latch will result in a stable set/clear state after a long enough time.

For eg., if one input of a NOR gate is 1, then after its propagation delay, its output will become 0, irrespective of the other input (even if that other input is floating).

Etc.

^ NAND-Gate D Flip-Flop

(https://youtu.be/YW-_GkUguMM)

Ben uses the RESISTOR-CAPACITOR method for EDGE-TRIGGERING.
However, the ICs that he later uses for D flip-flops are based on the undermentioned method for EDGE-TRIGGERING.

Let the PROPAGATION delay of every NAND gate be 1 unit, and the DURATION of a CLOCK CYCLE be 20 units, with the clock staying at HIGH & LOW for 10 units each and the clock transitioning from LOW to HIGH at t = 20 T, where T is an integer.

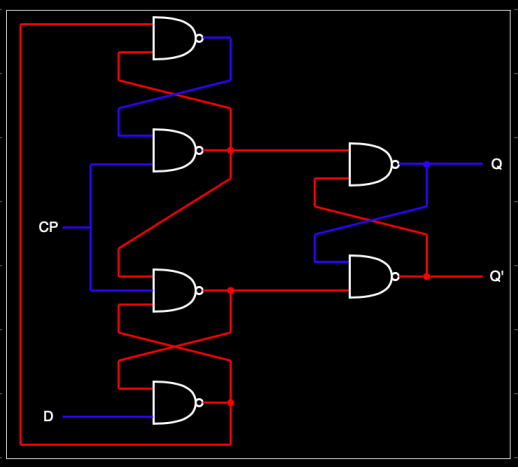
In real-world circuits, the duration of a clock cycle is MUCH MORE than this.

The SETUP time (i.e. the DURATION for which D must be CONSTANT while CP is at 0 IMMEDIATELY BEFORE CP gets changed from 0 to 1) is 2 units.
The HOLD time (i.e. the DURATION for which D must be CONSTANT while CP is at 1 IMMEDIATELY AFTER CP gets changed from 0 to 1) is 1 unit.
Hence, D must be constant from t = (20 T − 2) to t = (20 T + 1).
[These SETUP and HOLD times were calculated AFTER understanding the undermentioned transitions]

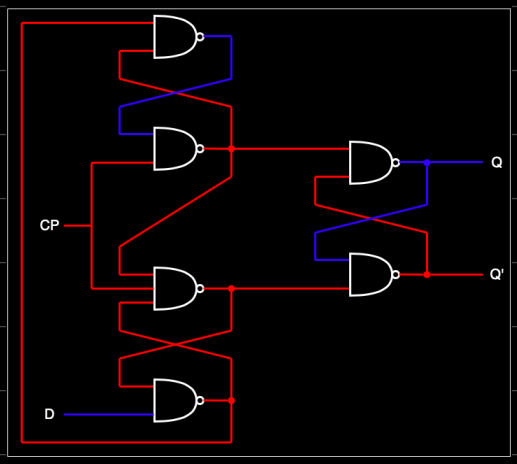
Let the initial STABLE state be CP = 0, D = 0, Q = 0 and Q' = 1 (CLEAR), and let the NEXT bit to be stored be 0 (which is present at the D input at t = 0 and whose SETUP time is completed), then 1, then 1, and then 0.

Let the next bit to be stored arrive at the D input WHILE the clock is STILL at 1 such that the HOLD time constraint is not violated, say at t = (20 T + 7).

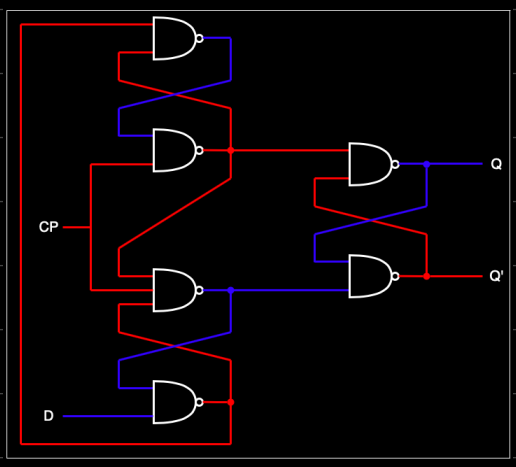
Till t = 0 :-



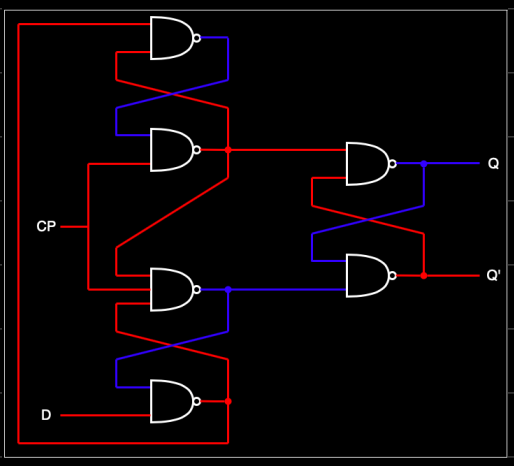
From t = 0 to t = 1 :-



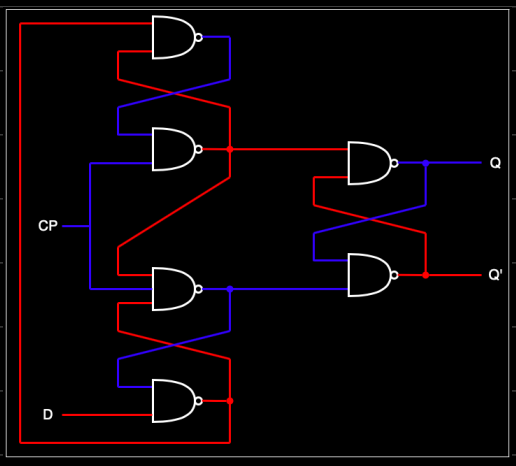
From t = 1 to t = 7 :-



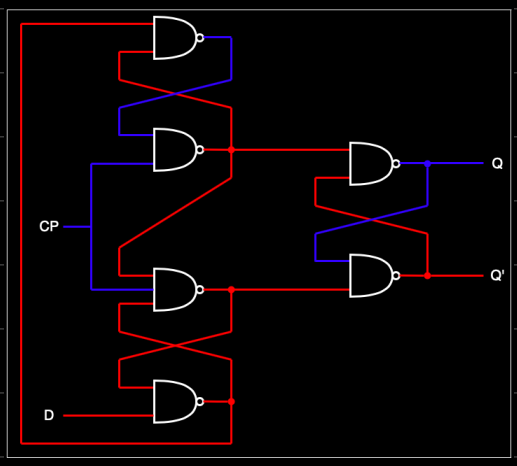
From t = 7 to t = 10 :-



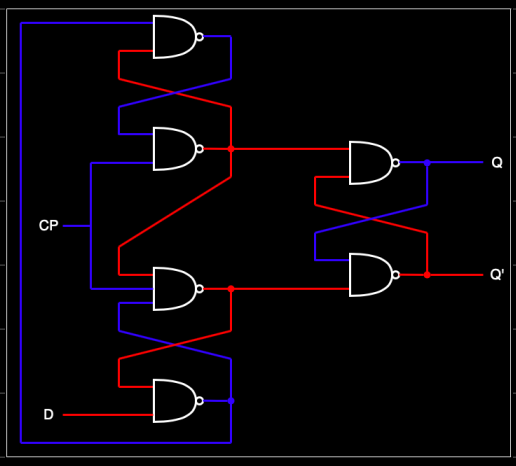
From t = 10 to t = 11 :-



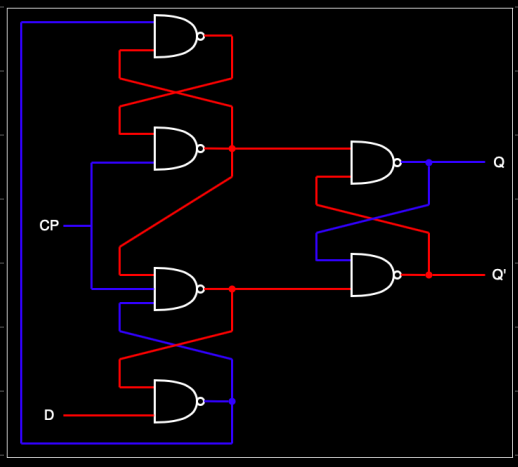
From t = 11 to t = 12 :-



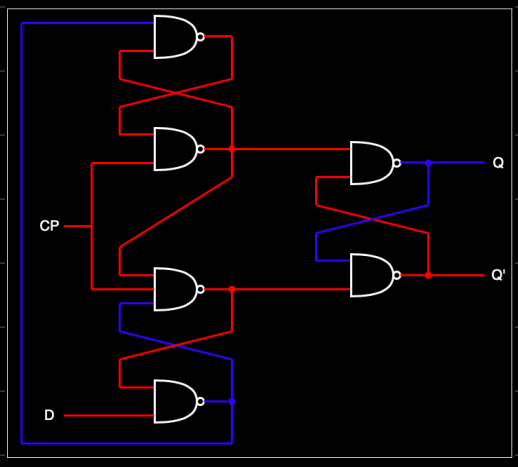
From t = 12 to t = 13 :-



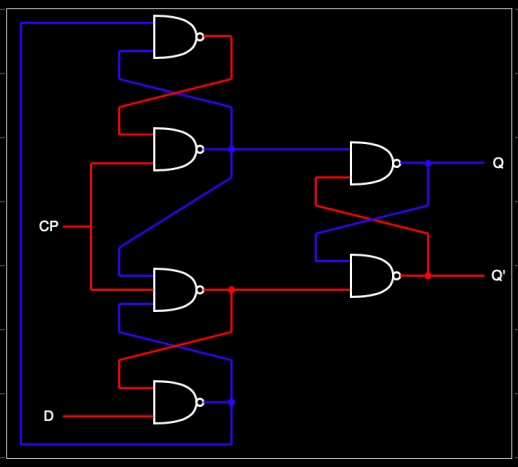
From t = 13 to t = 20 :-



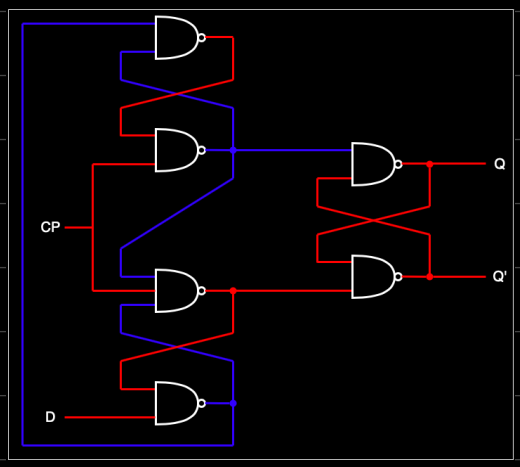
From t = 20 to t = 21 :-



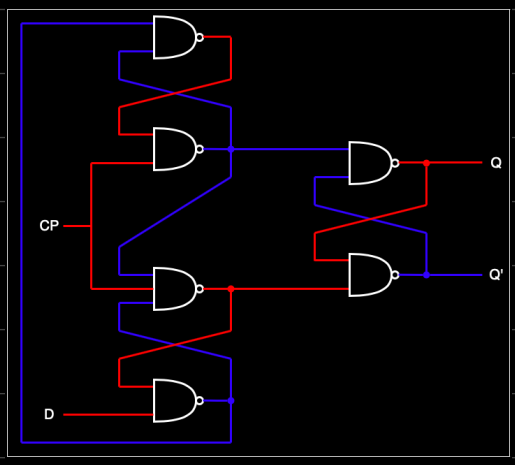
From t = 21 to t = 22 :-



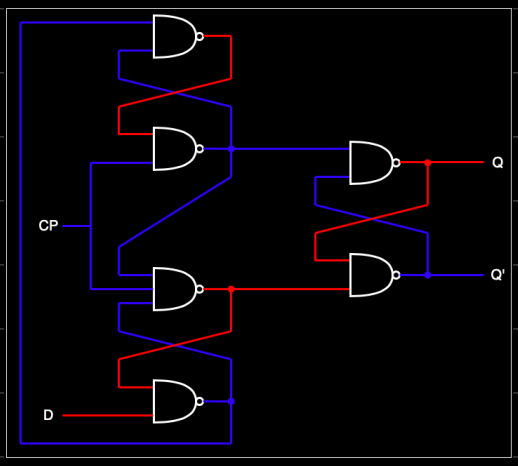
From t = 22 to t = 23 :-



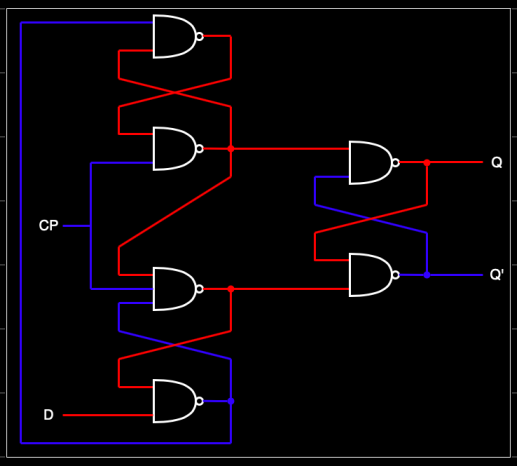
From t = 23 to t = 30 :-



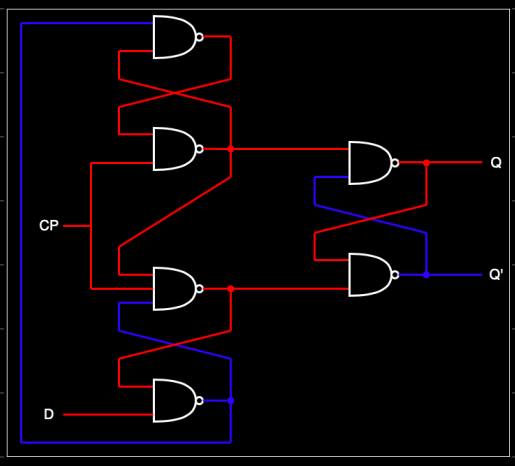
From t = 30 to t = 31 :-



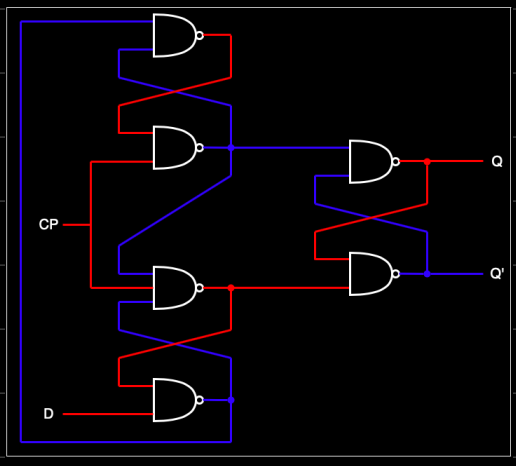
From t = 31 to t = 40 :-



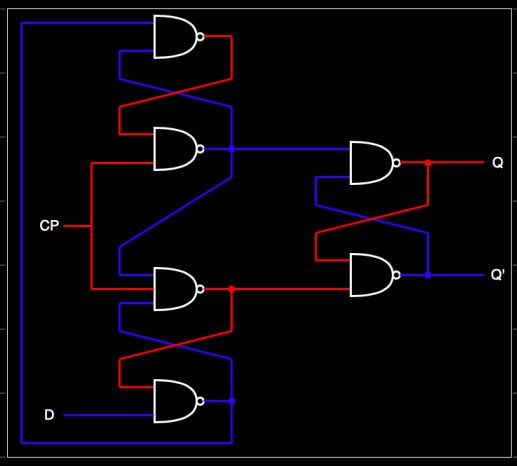
From t = 40 to t = 41 :-



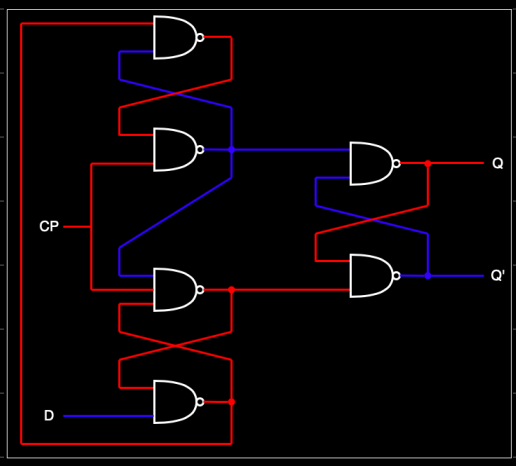
From t = 41 to t = 47 :-



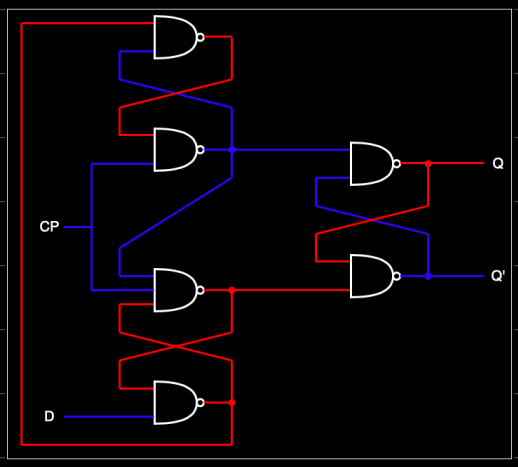
From t = 47 to t = 48 :-



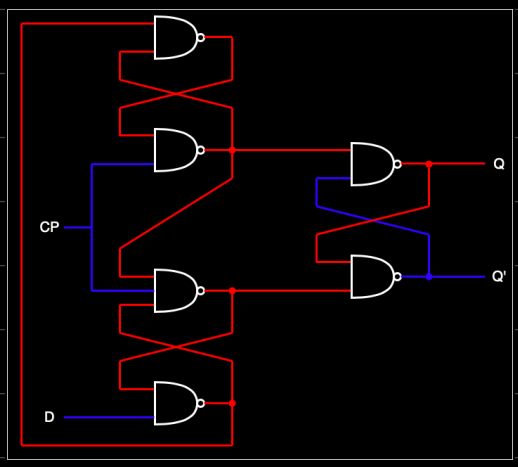
From t = 48 to t = 50 :-



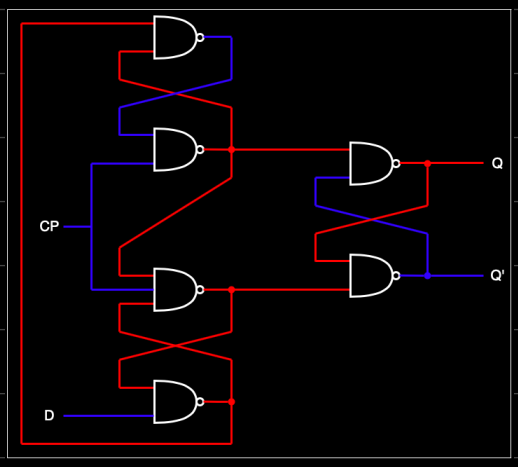
From t = 50 to t = 51 :-



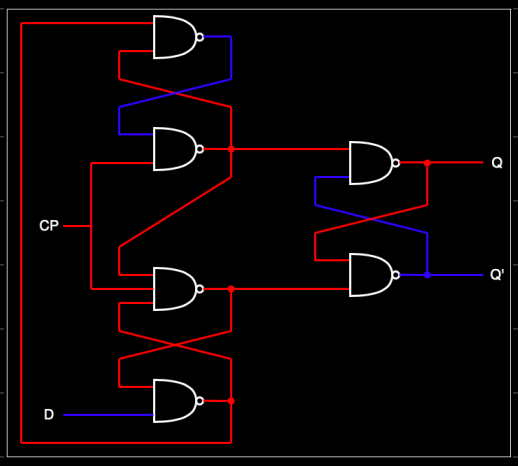
From t = 51 to t = 52 :-



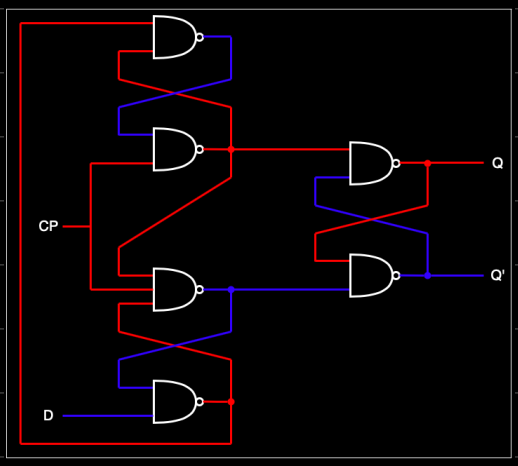
From t = 52 to t = 60 :-



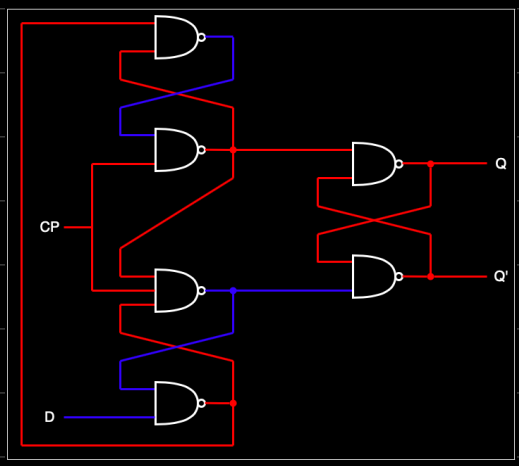
From t = 60 to t = 61 :-



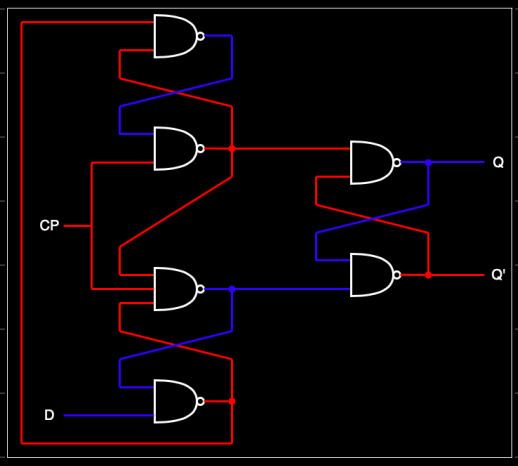
From t = 61 to t = 62 :-



From t = 62 to t = 63 :-



From t = 63 onwards :-

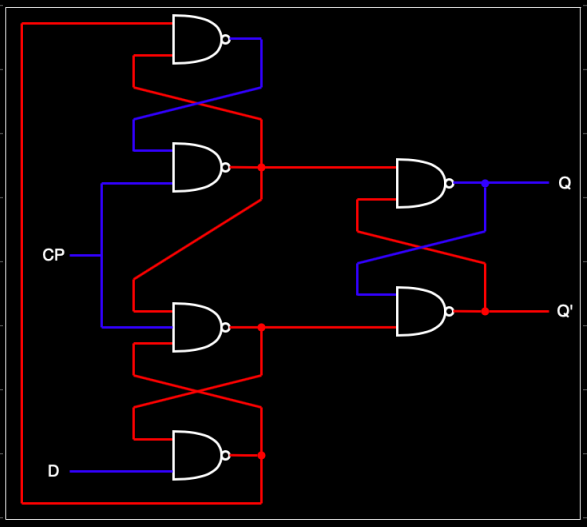


Explanation for the SETUP time and the HOLD time when the flip-flop is getting changed from CLEAR to SET :-

Let the initial STABLE state be

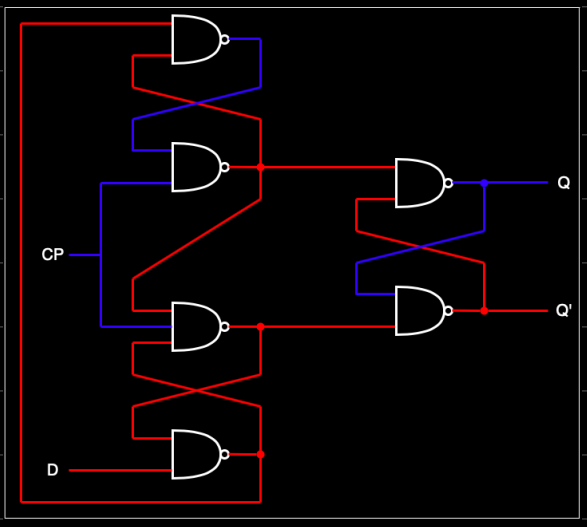
CP = 0, D = 0, Q = 0 and

Q' = 1 (CLEAR).

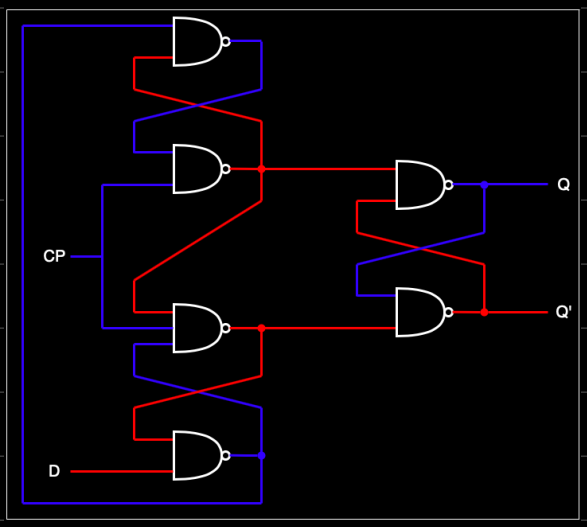


Let D get changed to 1 at t = 0.

From t = 0 to t = 1 –

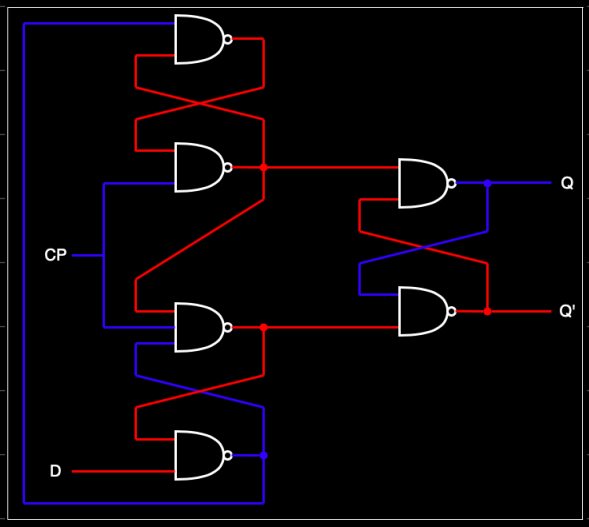


From t = 1 to t = 2 –



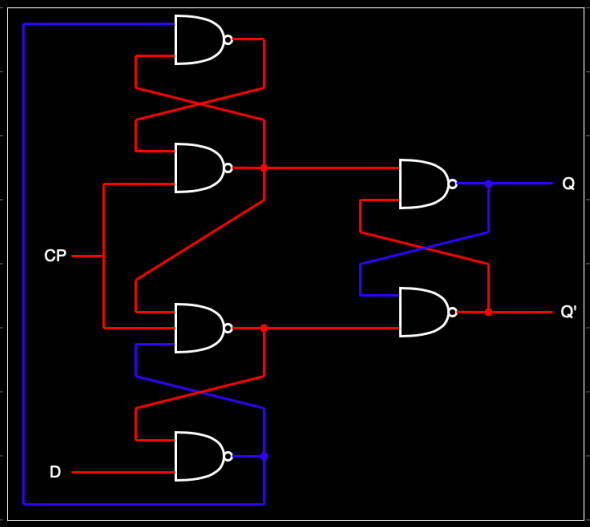
From t = 2 onwards –

So, the SETUP time is equal to the propagation delays through the BOTTOMMOST gate and then through the TOPMOST gate.



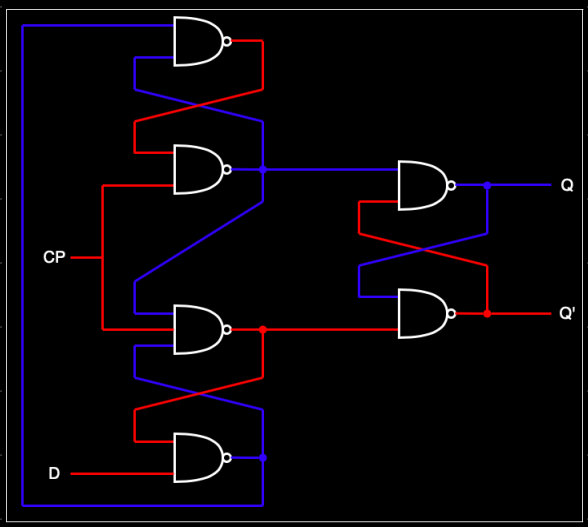
Now, let the initial STABLE state be CP = 0, D = 1, Q = 0 and Q' = 1 (CLEAR) and let CP get changed to 1 at t = 0.

From t = 0 to t = 1 –



From t = 1 to t = 2 –

So, changing D AFTER t = 1 will NOT affect the latch's next state, even if the propagation delays through the BOTTOMMOST gate and then through the TOPMOST gate are NEGLIGIBLE.



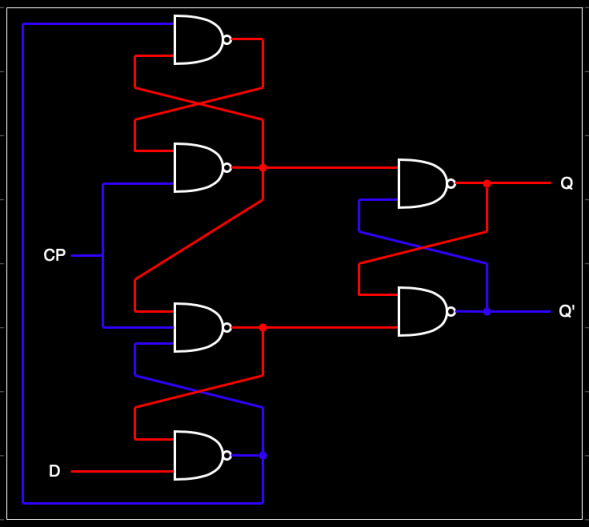
Therefore, the HOLD time is equal to the propagation delay through the UPPER gate to which CP is connected.

Explanation for the SETUP time and the HOLD time when the flip-flop is getting changed from SET to CLEAR :-

Let the initial STABLE state be

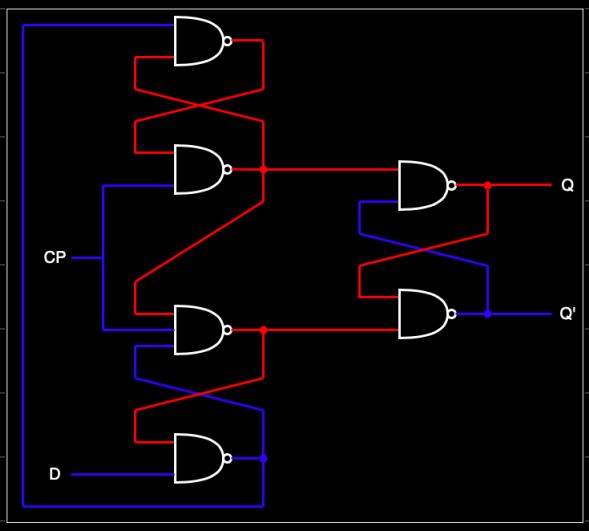
CP = 0, D = 1, Q = 1 and

Q' = 0 (SET).

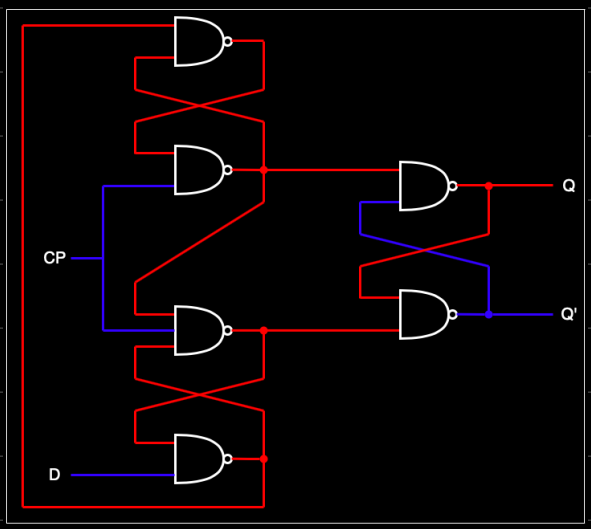


Let D get changed to 0 at t = 0.

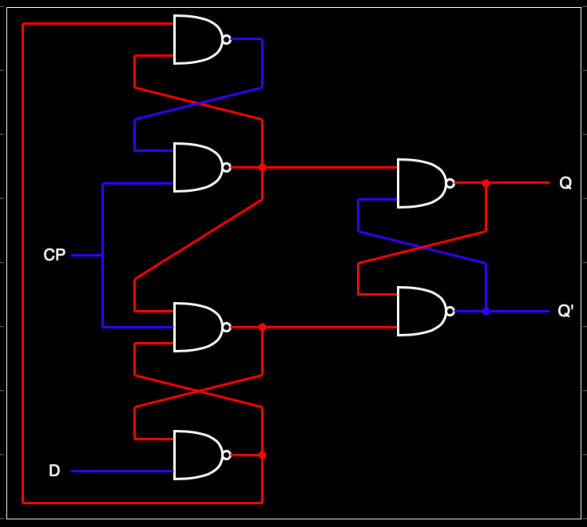
From t = 0 to t = 1 –



From t = 1 to t = 2 –

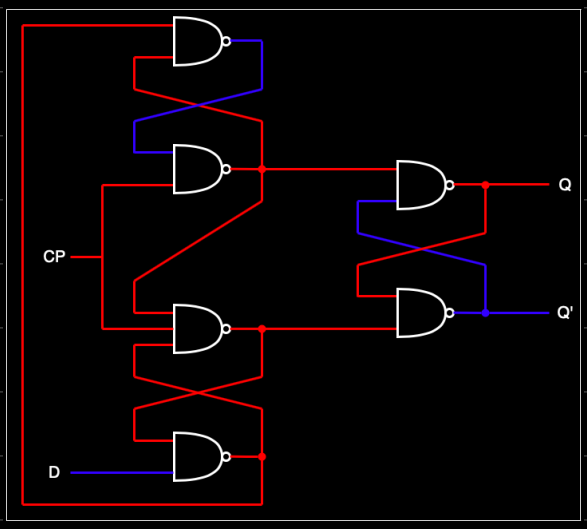


From t = 2 onwards –



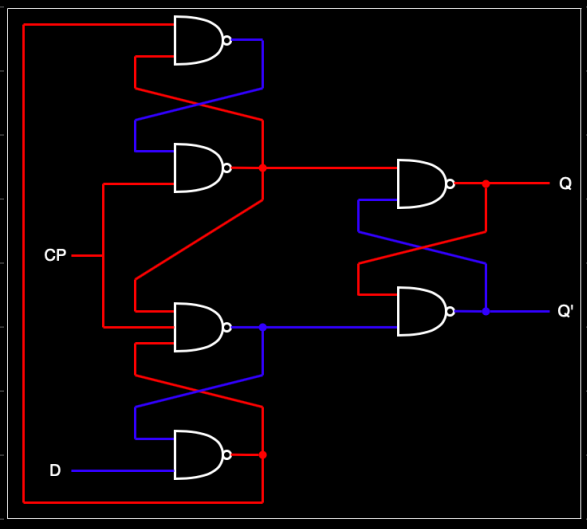
So, the SETUP time is equal to the propagation delays through the BOTTOMMOST gate and then through the TOPMOST gate. In the aforementioned transitions, from t = 50 onwards, it may SEEM like the SETUP time is 1 unit, but it should be noted that the FIRST part of SETUP, i.e. the propagation through the BOTTOMMOST gate, was already completed when CP was high.

Now, let the initial STABLE state be CP = 0, D = 0, Q = 1 and Q' = 0 (SET) and let CP get changed to 1 at t = 0.



From t = 0 to t = 1 –

From t = 1 to t = 2 –



So, changing D AFTER t = 1 will NOT affect the latch's next state, even if the propagation delay through the BOTTOMMOST gate is NEGLIGIBLE. Therefore, the HOLD time is equal to the propagation delay through the LOWER gate to which CP is connected.

In the computer to be developed in Ben Eater's playlist, and in GENERAL, the data to be loaded into gated latches / flip-flops in the NEXT clock cycle arrive at the inputs while the clock is LOW, while the clock was previously HIGH, etc., such that the SETUP time constraint is NOT violated.

Then, the data get loaded into those AFTER the clock goes HIGH in the NEXT clock cycle.

Finally, the inputs may get changed while the clock is still HIGH, after the clock goes LOW, etc., such that the HOLD time constraint is NOT violated.

These changed inputs MAY or MAY NOT correspond to the data to be loaded in the future.

To be answered later (?) :-

What happens if D is not maintained constant during the setup time?

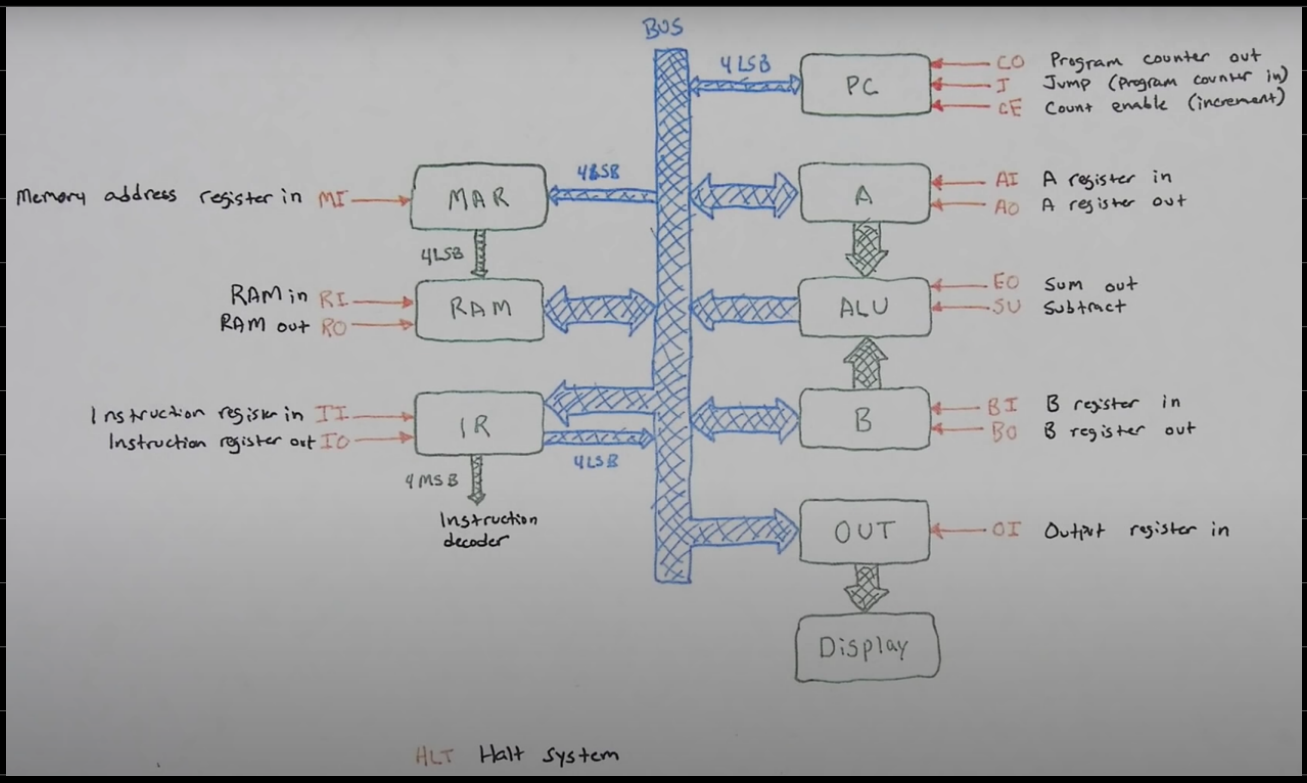
What happens if CP gets changed to 1 before the setup time is completed?

What happens if D is not maintained constant for the hold time?

What happens if CP is not maintained at 1 for a long enough time?

Etc.

^ Microinstructions



There are a total of 16 control signals, forming a 16-bit CONTROL WORD.

From the above figure, BO is not used, and instead, FI is used.

The constituent LOGIC GATES of the ICs that Ben uses are constructed/designed to convert disconnected inputs and floating inputs to 1.

So, AFTER the computer is powered up and BEFORE the RESET button is pressed, the OUTPUTS of all the GATED LATCHES / FLIP-FLOPS will have ARBITRARY values (i.e. STABLE 0, STABLE 1 or OSCILLATING between 0 & 1, but NOT floating) which MAY cause arbitrary CONTROL WORDS to get generated.

Moreover, the CLOCK input will stay at 0 during this time.

The BUS is constructed/designed such that the values on its wires become STABLE 0's when the bus is NOT being driven. Thus, those INPUTS to the logic gates which are coming from the BUS will NEVER be FLOATING.

However, there would be NO problems EVEN IF the values on the wires of the bus would be FLOATING when the bus is NOT being driven.
This is because, for eg., if ONE input of a NOR gate is 1, then after its PROPAGATION delay, its output will become 0, IRRESPECTIVE of the OTHER input (even if that other input is FLOATING). Similarly, the CLOCK input or the IN signal of a component being DISABLED would determine the output(s) of the corresponding logic gate(s) IRRESPECTIVE of the input(s) coming from the BUS (even FLOATING).

the reset button is pressed which clears (STABLE 0)
ALL of the components, EXCEPT for the contents of the RAM.
The constituent LOGIC GATES of the ICs that Ben uses are constructed/designed to convert disconnected inputs and floating inputs to 1. Hence, for every gated D latch of the RAM, the output will NOT remain FLOATING, and may become a STABLE 0, a STABLE 1 or OSCILLATING BETWEEN 0 & 1.

When required, the contents of the RAM will be written to.
Doing this will NOT cause a problem EVEN IF the outputs of some gated D latches of the RAM remained FLOATING because, for eg., for the NOR-gate SR latch, after S and R become CONSTANT at 1/0 and 0/1, respectively, then NO MATTER the current state of the latch, STABLE, OSCILLATING or even FLOATING, the latch will result in a STABLE SET/CLEAR state after a LONG ENOUGH time.

Before the RESET button is pressed, the INSTRUCTION REGISTER, the STEP COUNTER and the FLAGS REGISTER may have ARBITRARY values which MAY cause arbitrary MICROINSTRUCTIONS to get executed.

The time DURATION for which the RESET button is pressed is LONG ENOUGH for the signals to PROPAGATE through the RESET circuit and then through the individual ICs.
AFTER the RESET button is released and BEFORE the program to be run is MANUALLY loaded into the RAM, a LONG ENOUGH time passes for the RESET inputs on the individual ICs to get DISABLED after the signals getting PROPAGATED through the RESET circuit.

The COUNTERS (program counter and step counter) are POSITIVE edge-triggered, i.e. for such a counter, the MASTER gated latches work when the CLOCK input is at 0, and the SLAVE gated latches work when the CLOCK input is at 1.

So, for every master-slave flip-flop of such a counter, the CLOCK input goes directly to the SLAVE latch, and it goes through an internal INVERTER to the MASTER latch.

For the PROGRAM COUNTER, before the RESET button is pressed, the CLOCK inputs of every MASTER latch become 1 after the signals get PROPAGATED through the INVERTERS.

after it gets reset and its RESET input gets DISABLED, its

CALL THESE INTERNAL AND EXTERNAL INVERTERS.

CLOCK input is at 0. But, its COUNT ENABLE is at 0 as well. So, the ou

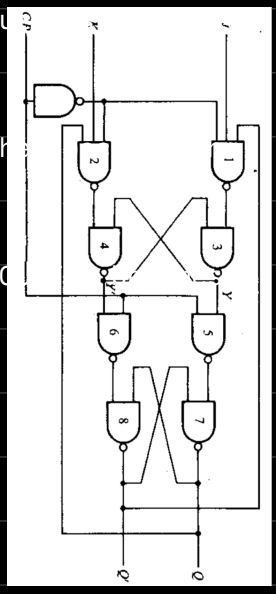
MASTER gated latch and every SLAVE gated latch stay at 0.

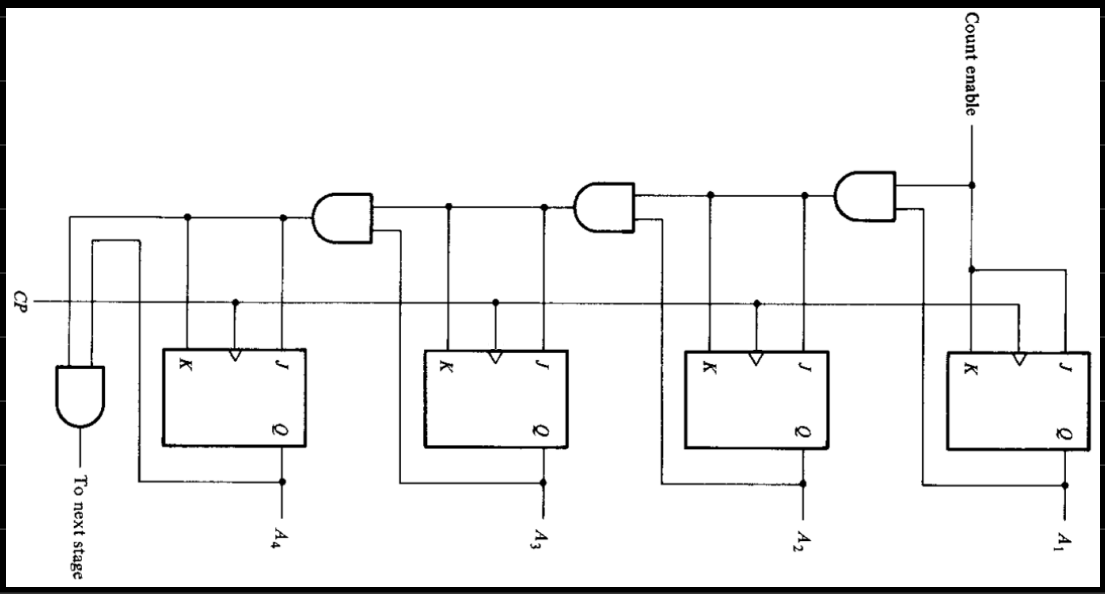
The CLOCK input of the PROGRAM COUNTER goes through an INVERTER and th

CLOCK input of the STEP COUNTER.

So, before the RESET button is pressed, the CLOCK input of the STEP CO

1 after the signal gets PROPAGATED through the INVERTER.





Now, for the STEP COUNTER, after it gets reset and its RESET input gets DISABLED, its CLOCK input is at 1. So, even though its COUNT ENABLE is always at 1, the outputs of every MASTER gated latch and every SLAVE gated latch stay at 0.