(*) Building an 8-bit Breadboard Computer

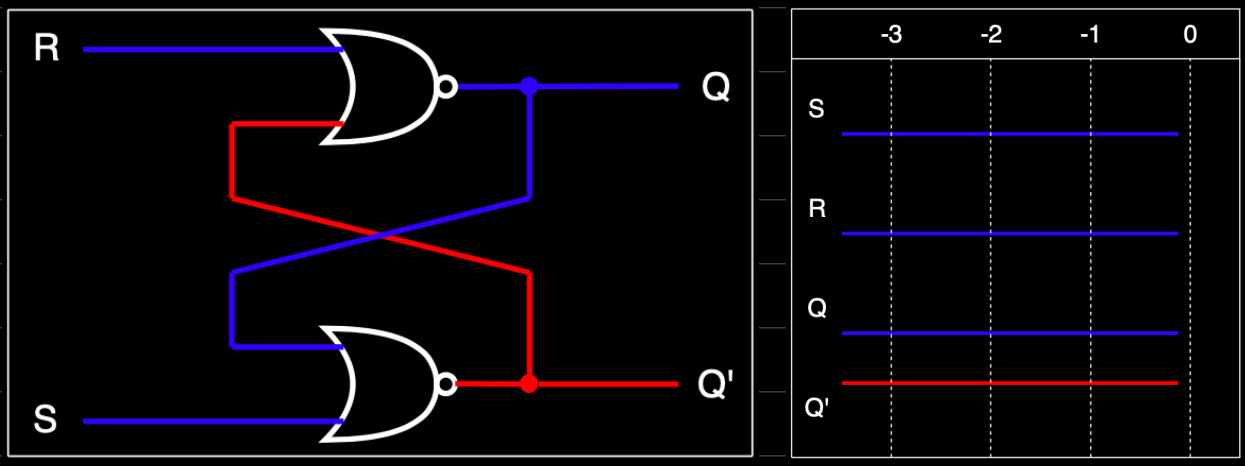
by Ben Eater

(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.

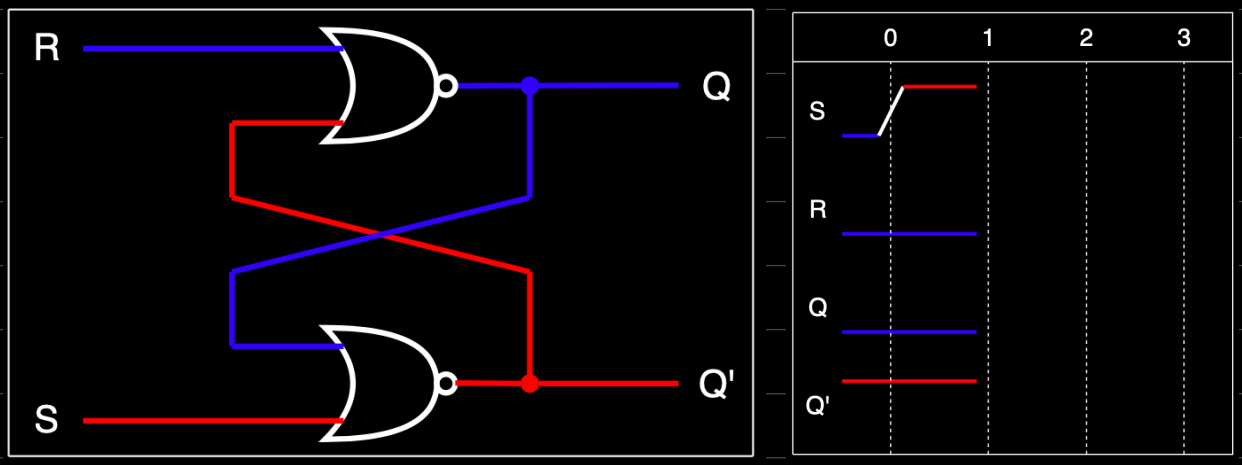
So, the EFFECTS of the changed inputs will be visible only AFTER the propagation delays.

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.

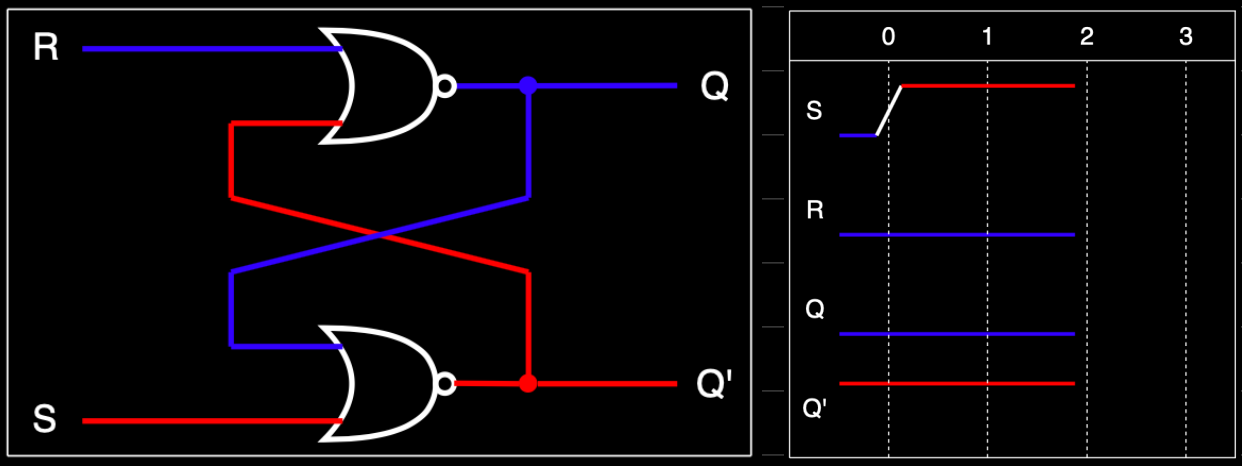
For eg., if the propagation delay of a logic gate is 3 units, then its OUTPUT at t = 3.5 units (say) depends upon its INPUTS at t = 0.5 units.

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

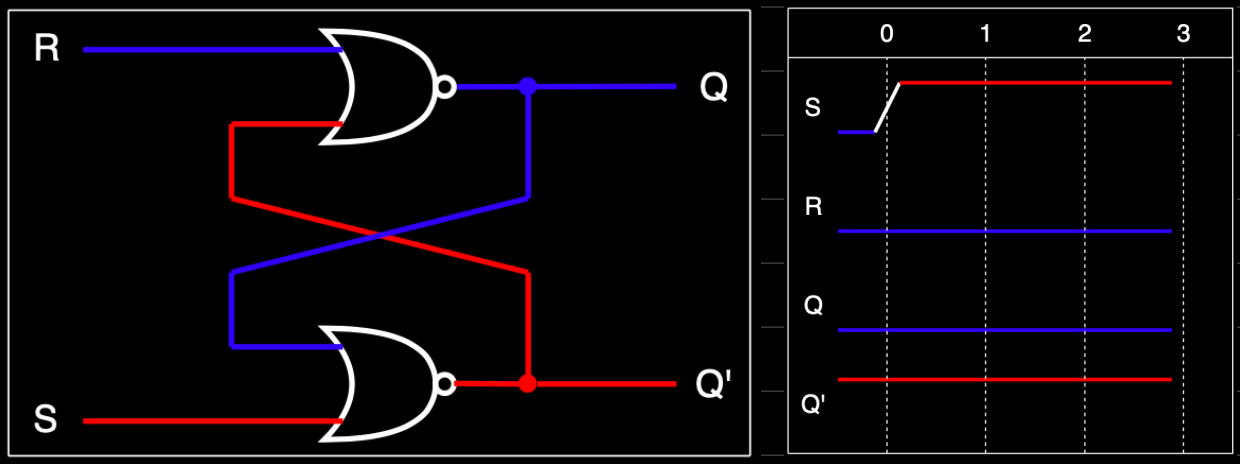
From t = 0 to t = 1 :-



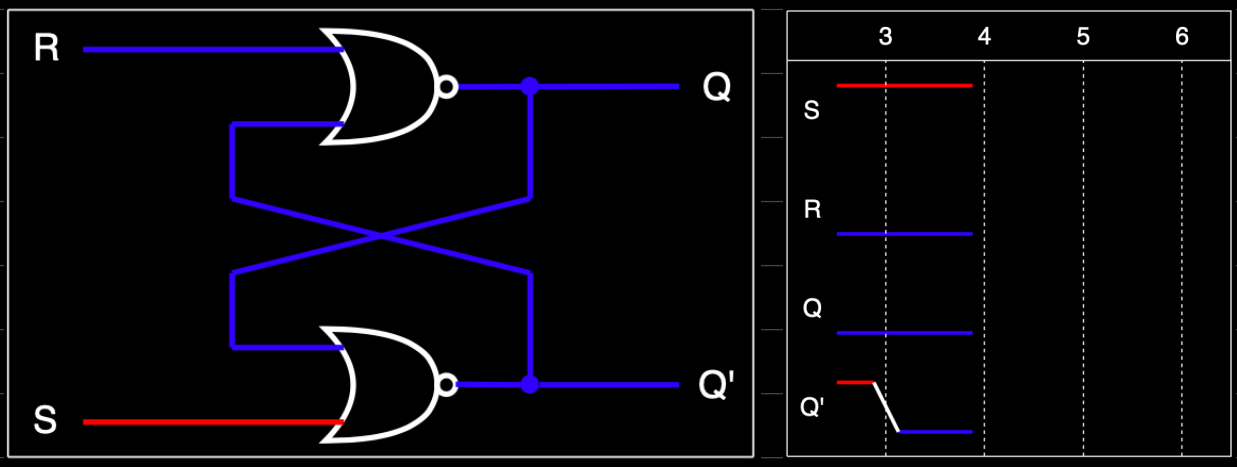
From t = 1 to t = 2 :-



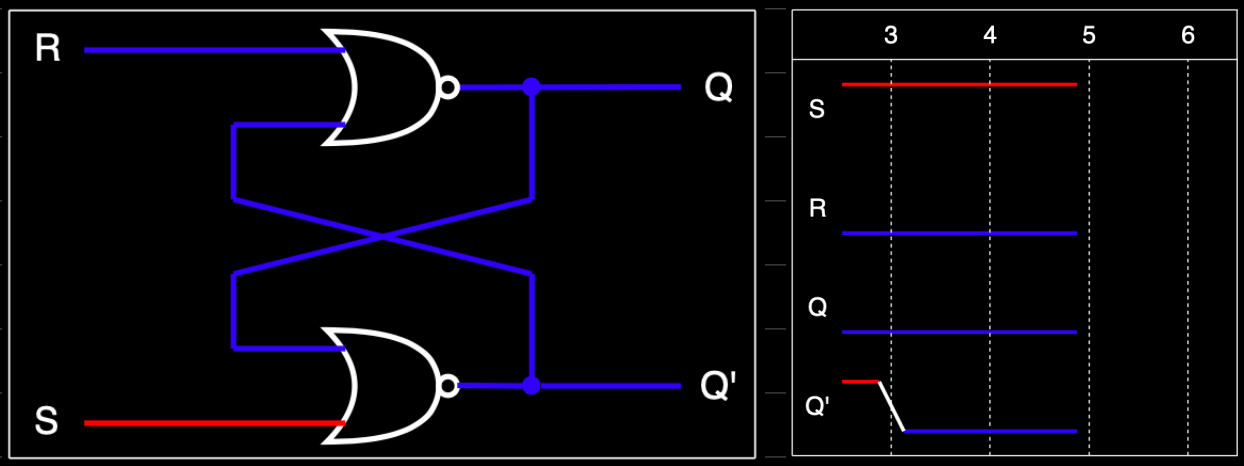
From t = 2 to t = 3 :-



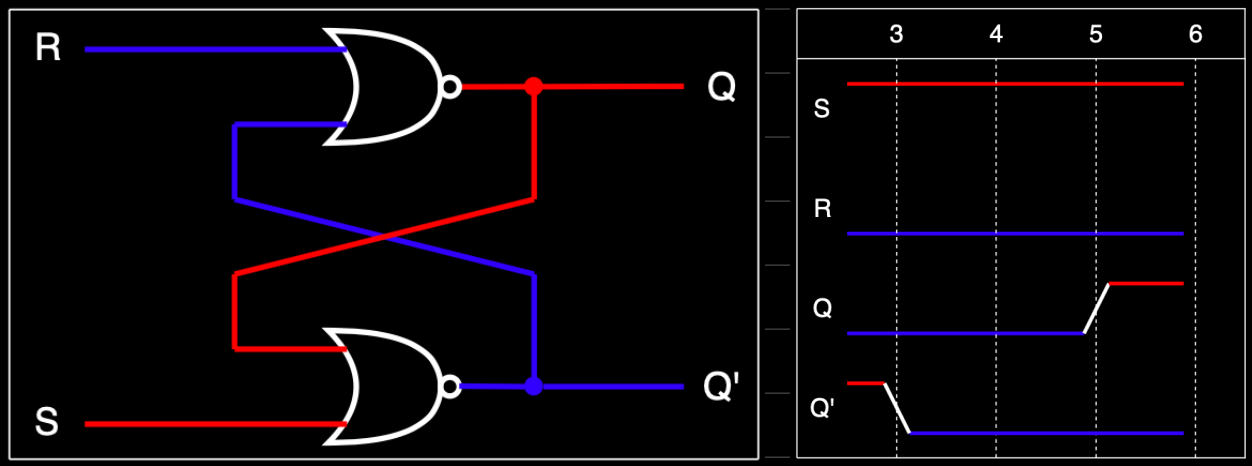
From t = 3 to t = 4 :-



From t = 4 to t = 5 :-



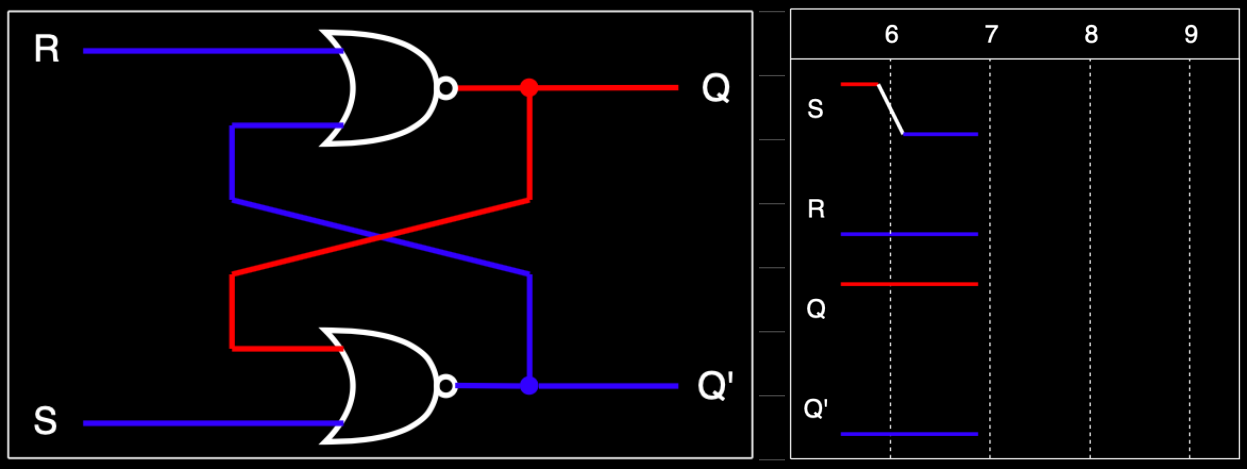
From t = 5 to t = 6 :-



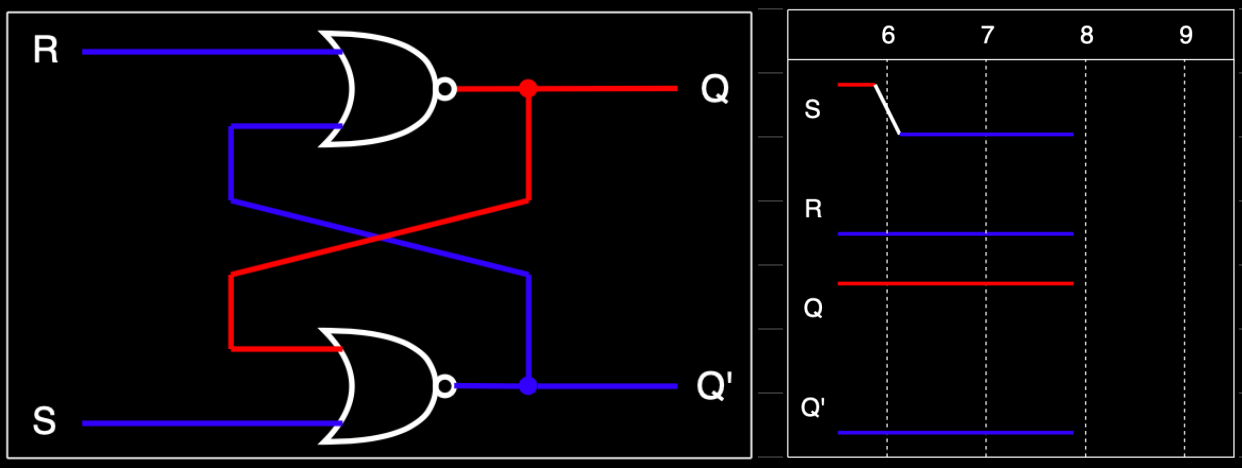
Now, the latch will stay STABLE in the SET state, even if S gets changed back to 0.

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

From t = 6 to t = 7 :-



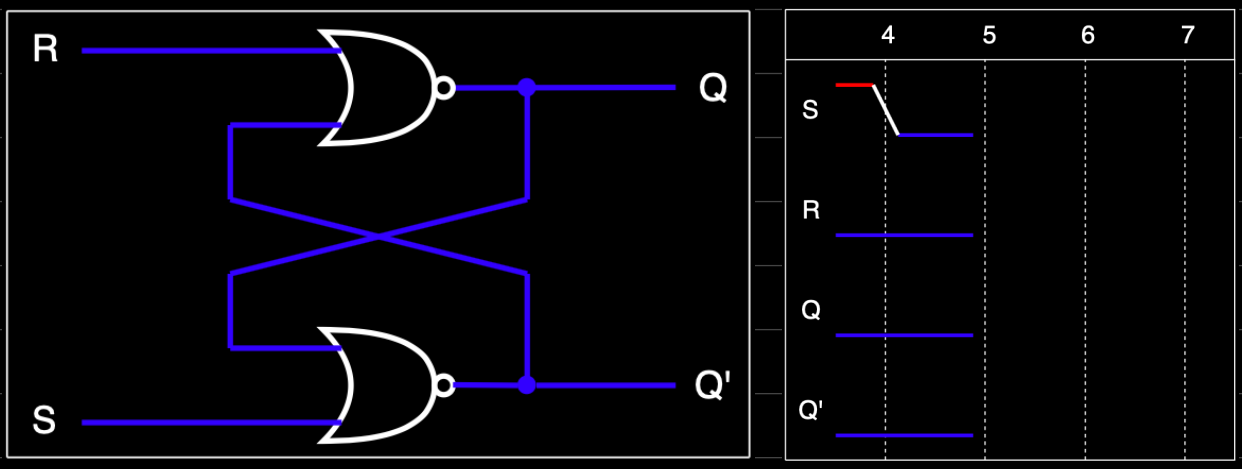
From t = 7 to t = 8 :-



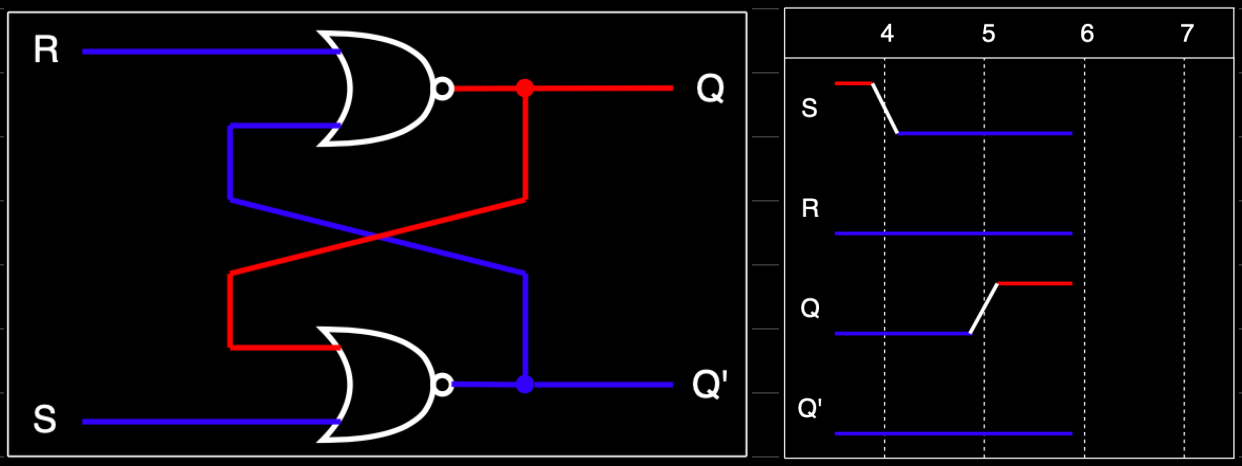
Hence, the latch will stay STABLE in the SET state, as long as S was maintained at 1 for a LONG ENOUGH time, i.e. if S gets changed back to 0 AFTER t = 5, then the latch will result in a STABLE SET state.

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

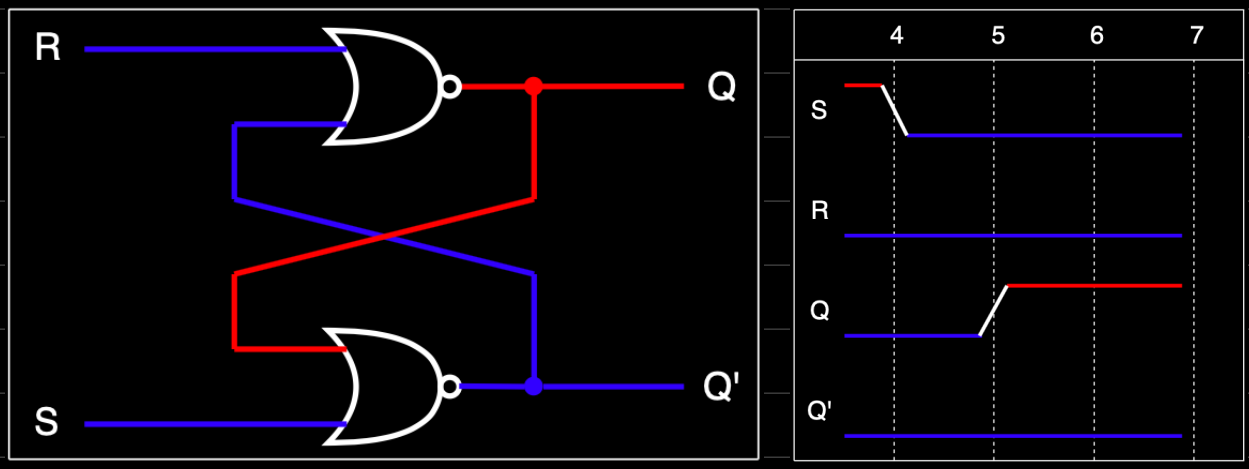
From t = 4 to t = 5 :-



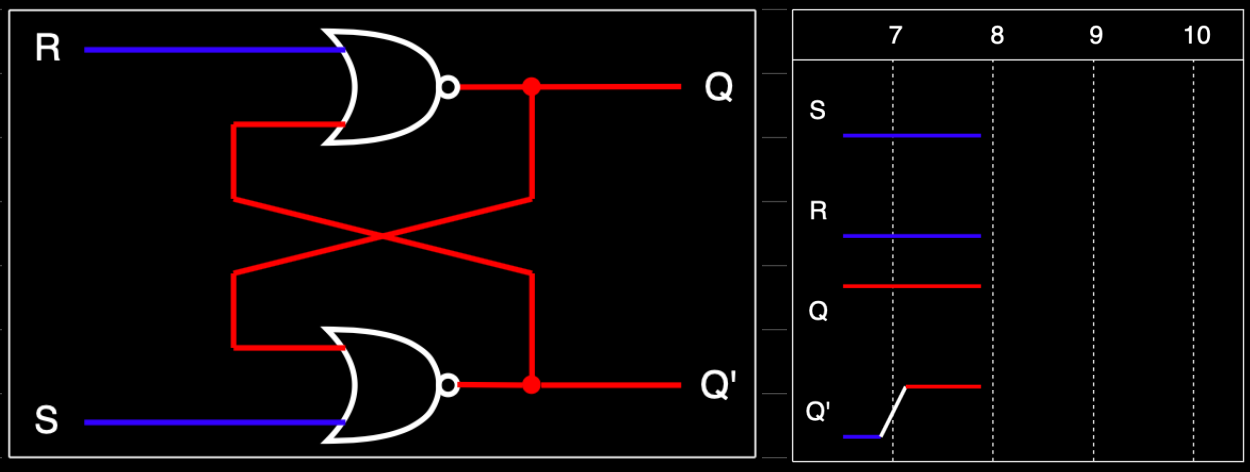
From t = 5 to t = 6 :-



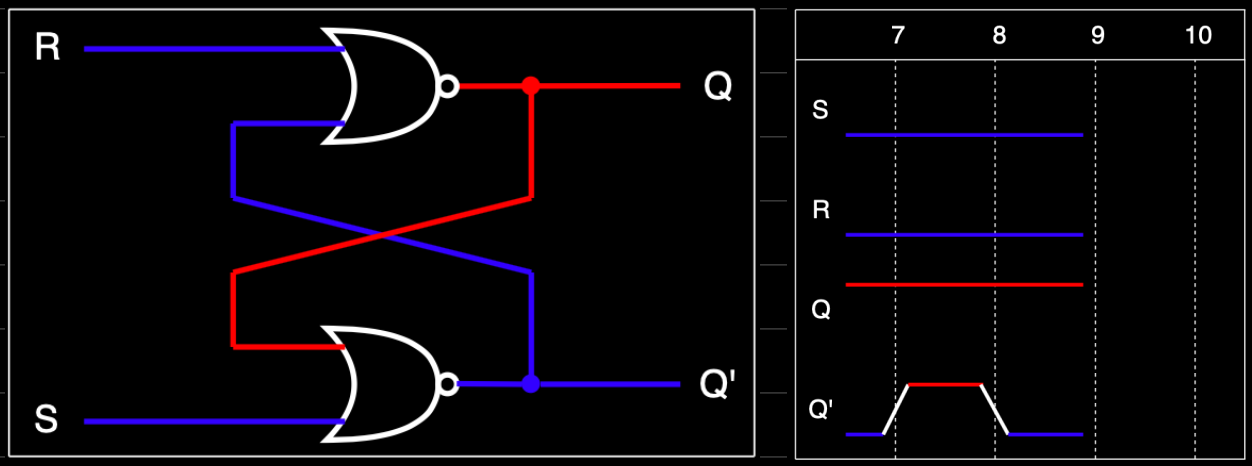
From t = 6 to t = 7 :-



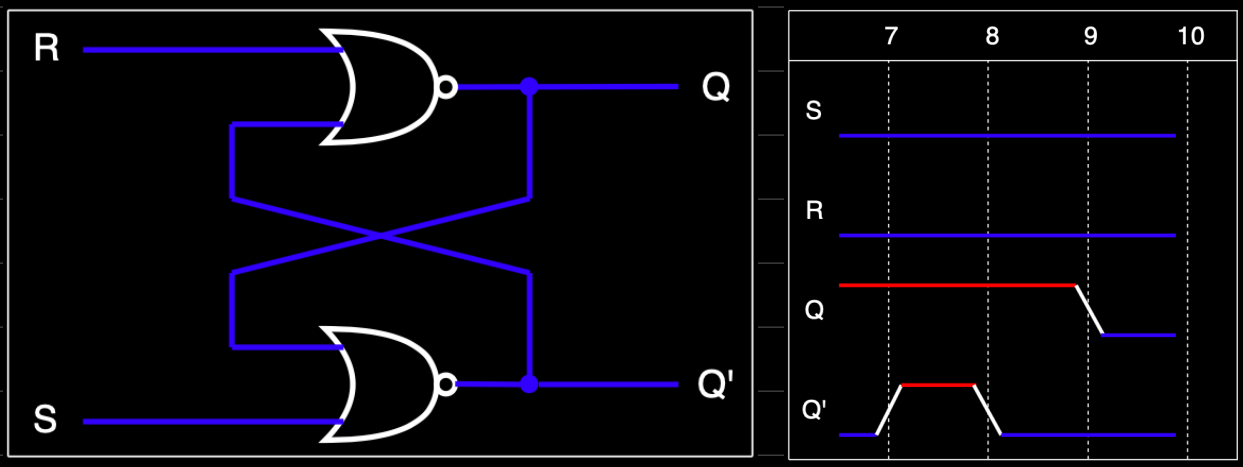
From t = 7 to t = 8 :-



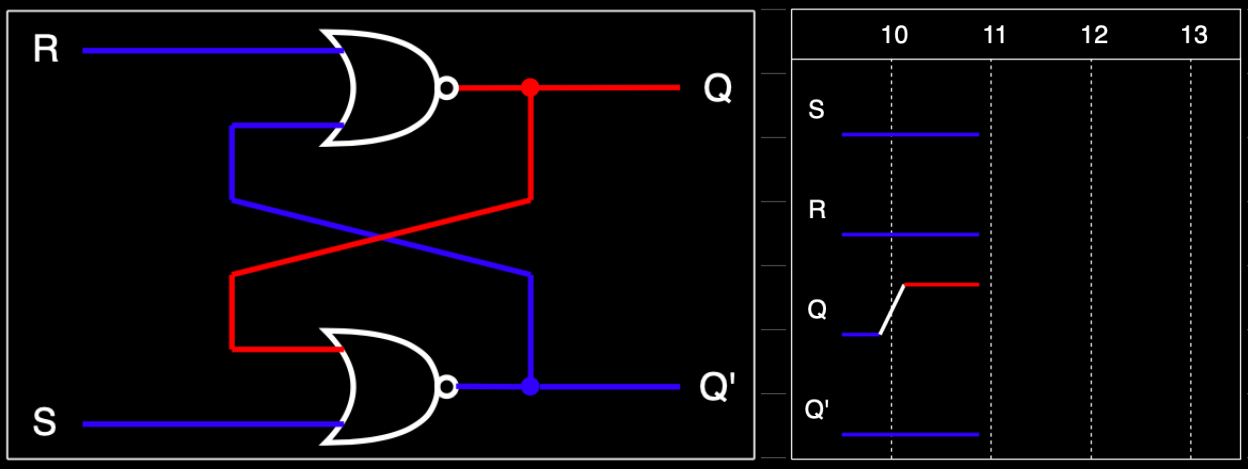
From t = 8 to t = 9 :-



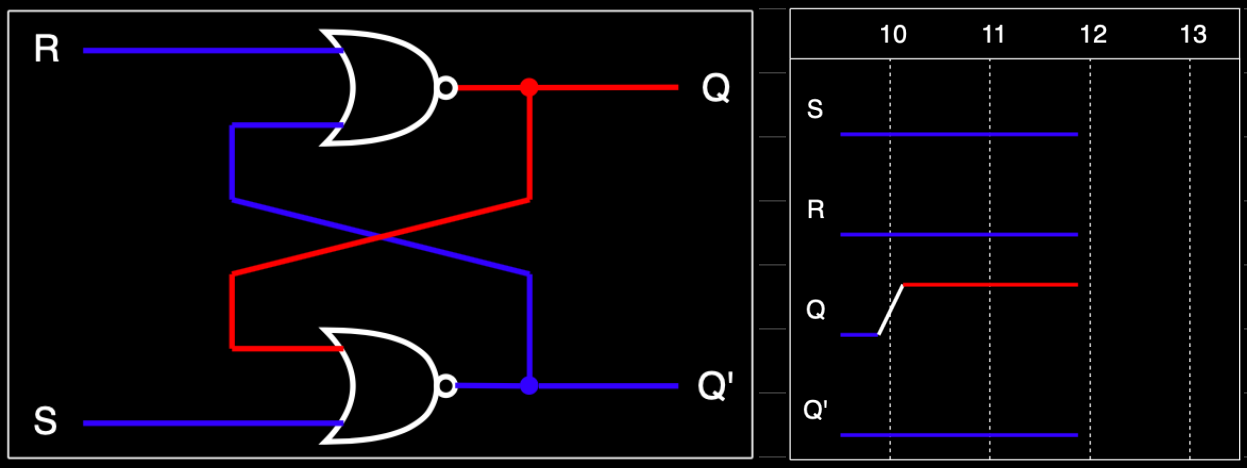
From t = 9 to t = 10 :-



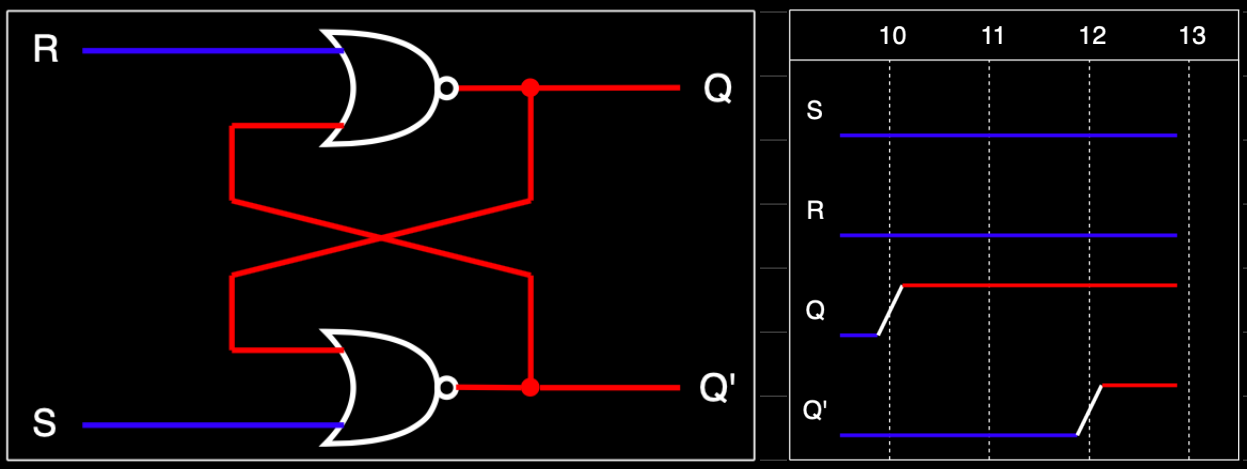
From t = 10 to t = 11 :-



From t = 11 to t = 12 :-



From t = 12 to t = 13 :-

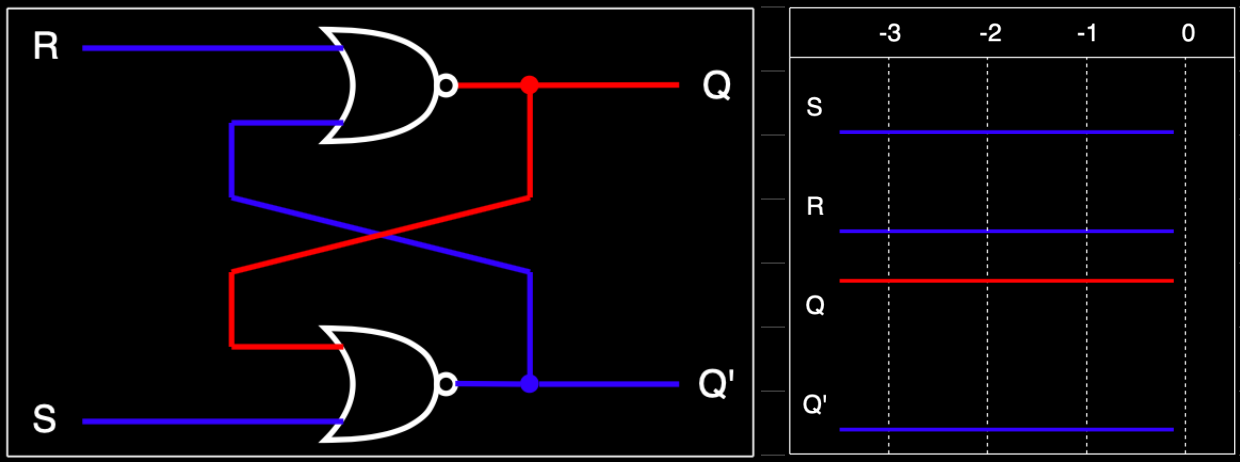


The latch will keep oscillating like this FOREVER. Previously, when the latch was being SET, it temporarily switched to an INVALID state, i.e. Q = Q' = 0, but then it became STABLE in the SET state. But, in this case, the latch will NEVER become STABLE in ANY state. Therefore, if S is NOT maintained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

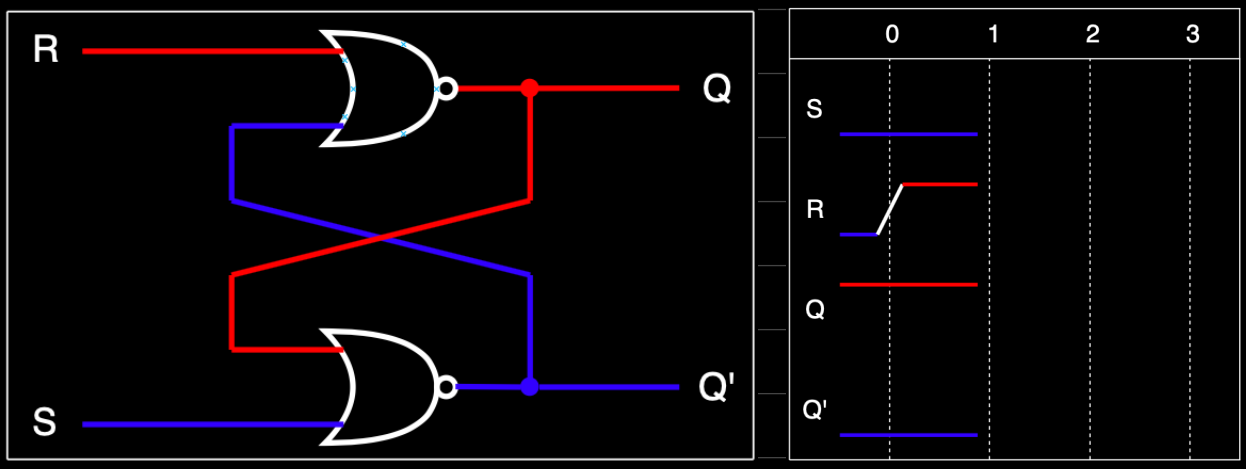
In any case, if used CORRECTLY, i.e. by keeping S/R high for a LONG ENOUGH time when setting/clearing the latch and by NOT keeping S & R high at the same time, then the latch will work AS EXPECTED in ideal as well as real-world circuits.

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

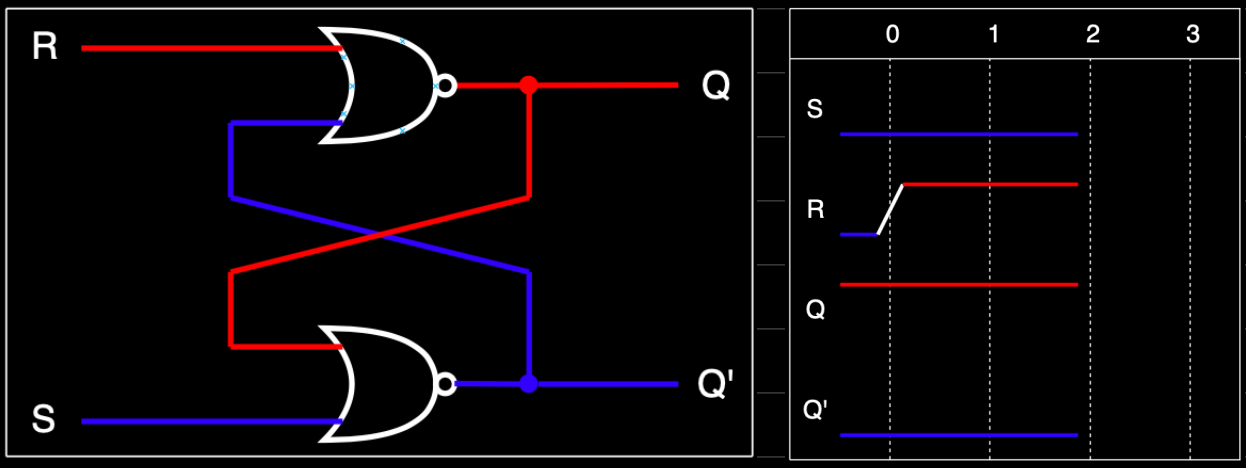


Let R get changed to 1 at t = 0 (in order for the latch to become CLEAR).

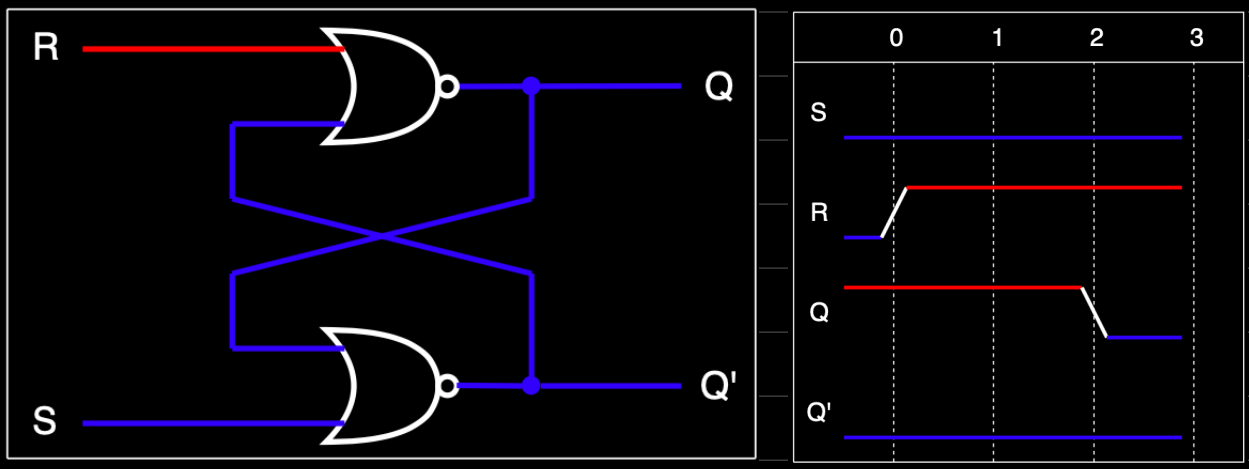
From t = 0 to t = 1 :-



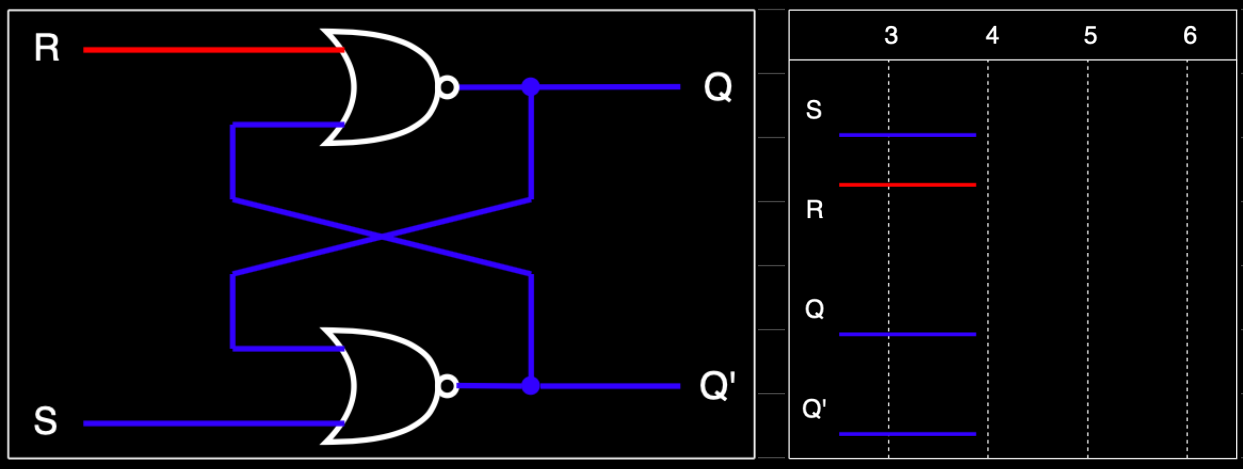
From t = 1 to t = 2 :-



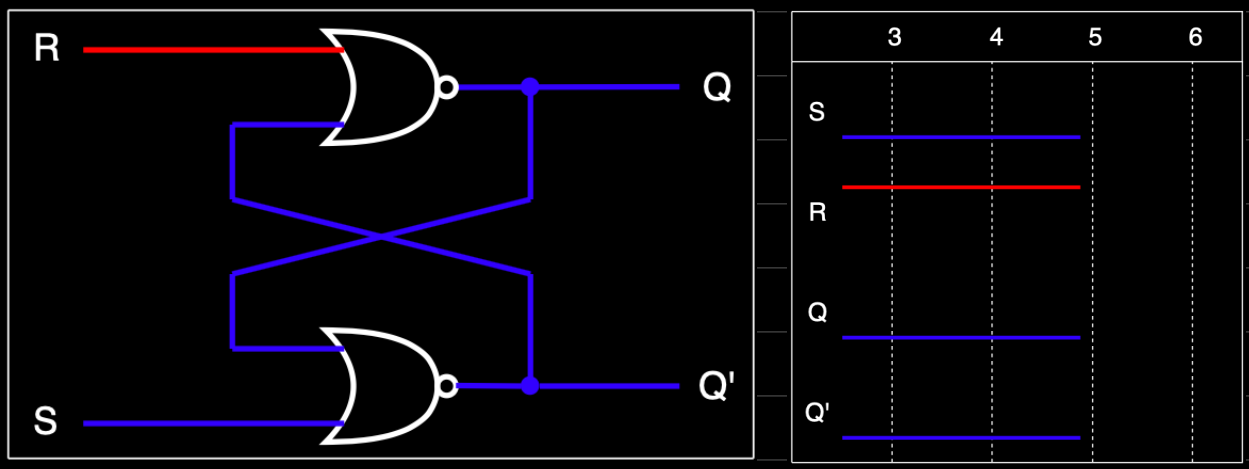
From t = 2 to t = 3 :-



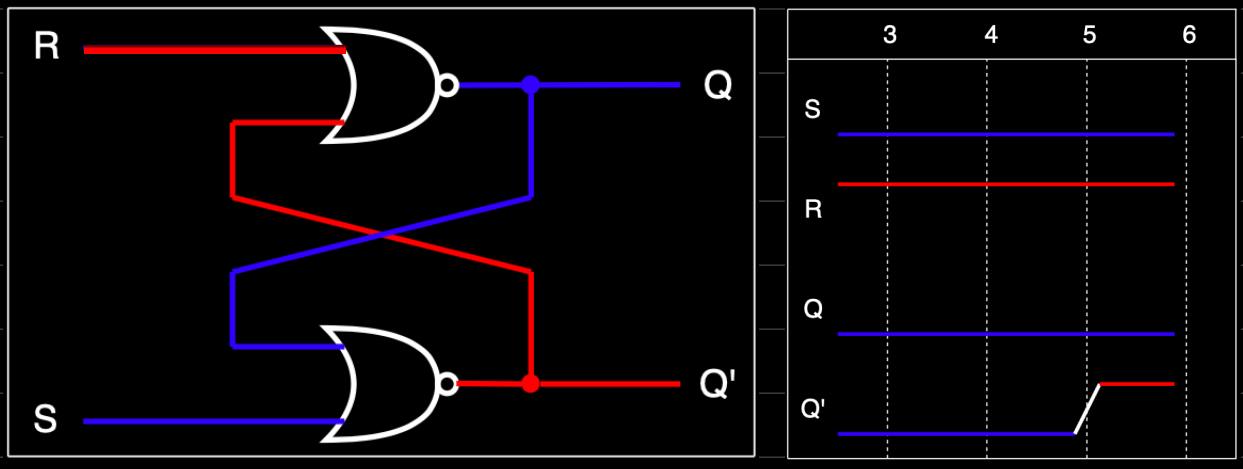
From t = 3 to t = 4 :-



From t = 4 to t = 5 :-



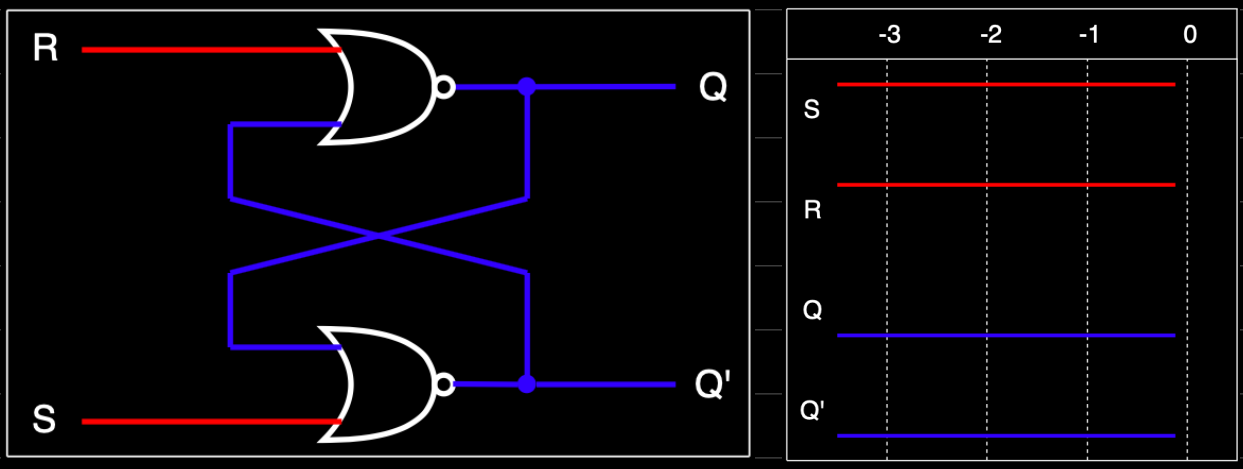
From t = 5 to t = 6 :-



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.

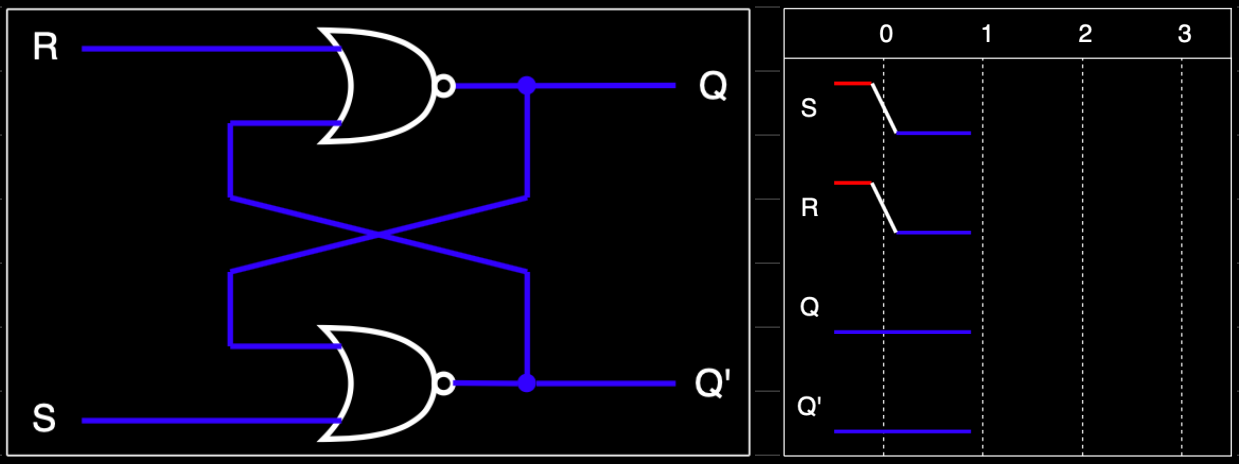
Therefore, if R is NOT maintained at 1 for a LONG ENOUGH time, then similar problems may occur like if S is NOT maintained at 1 for a LONG ENOUGH time when SETTING the latch.

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

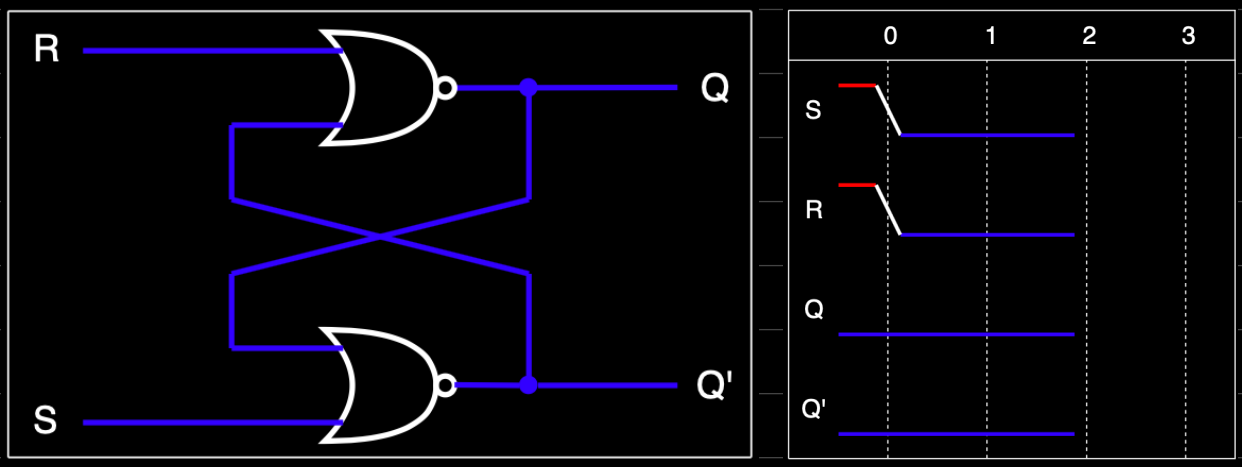


Let S & R both get changed to 0 at t = 0.

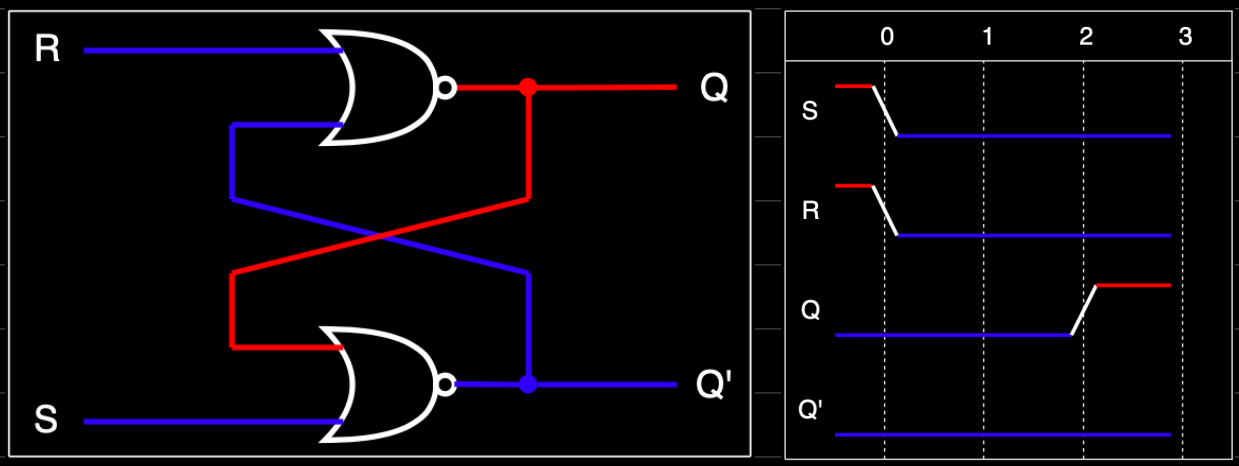
From t = 0 to t = 1 :-



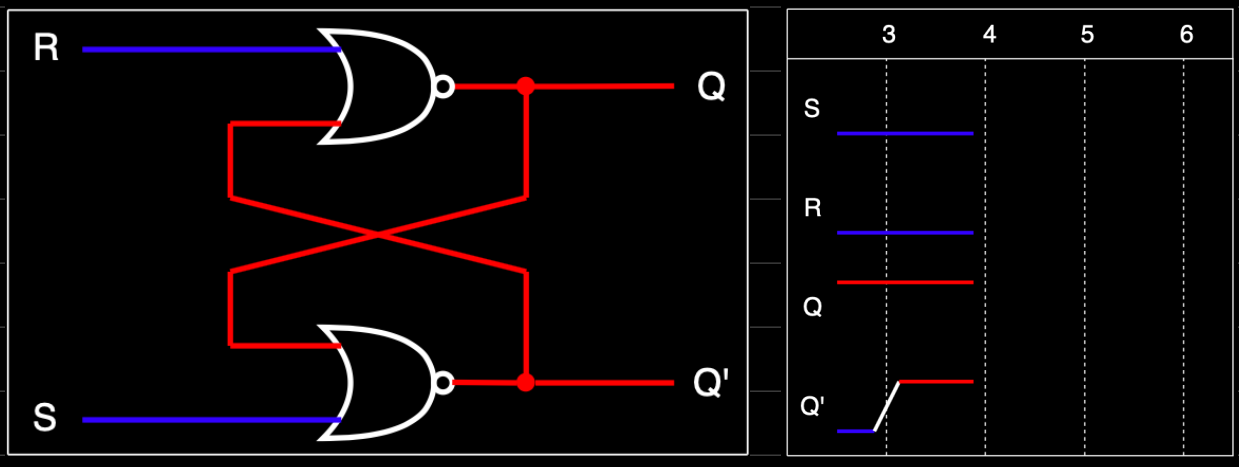
From t = 1 to t = 2 :-



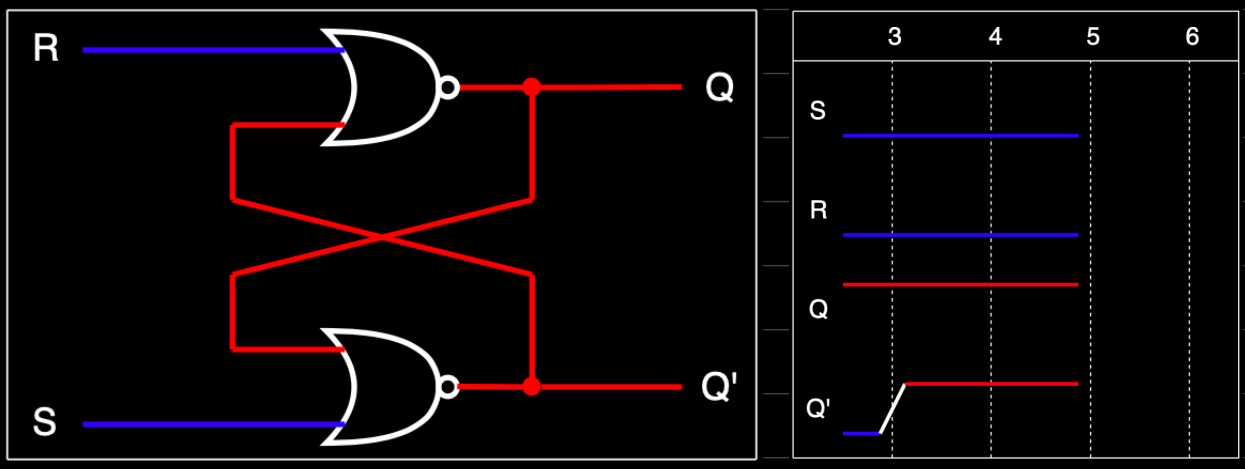
From t = 2 to t = 3 :-



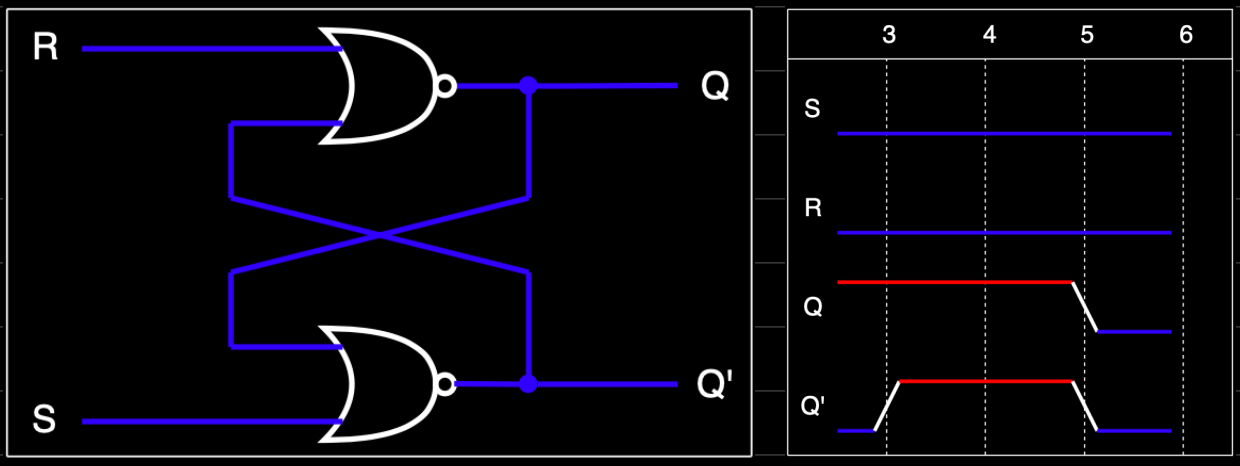
From t = 3 to t = 4 :-



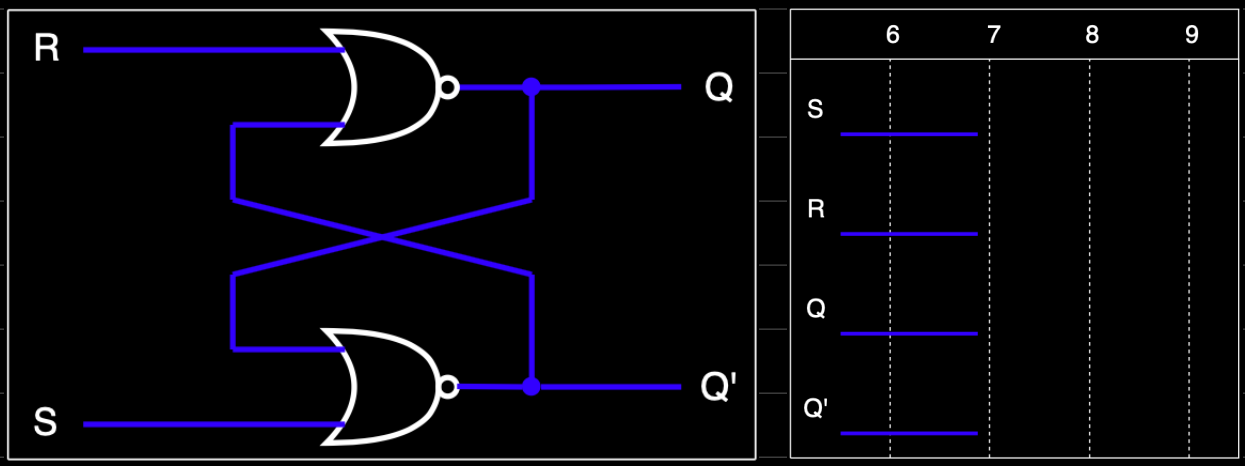
From t = 4 to t = 5 :-



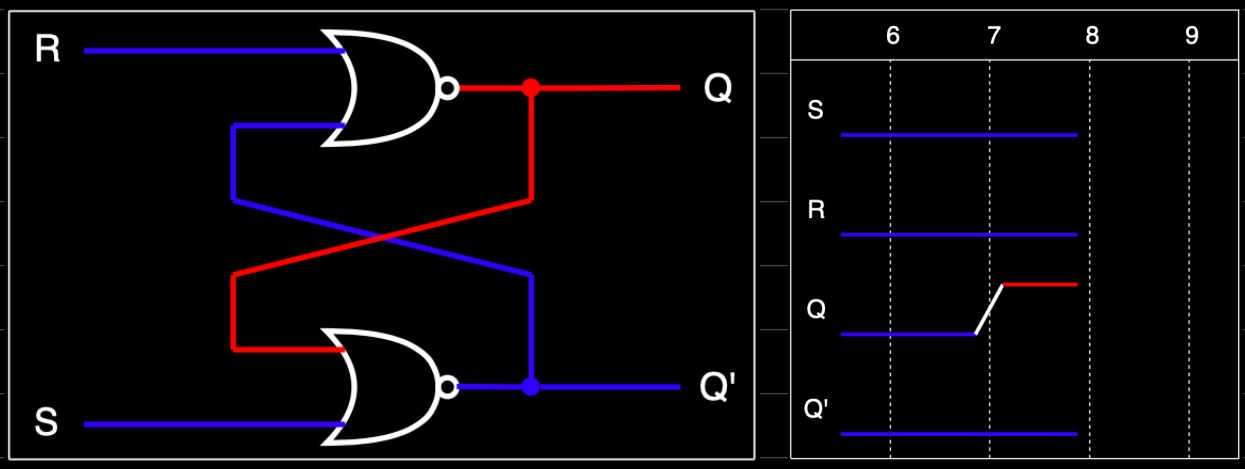
From t = 5 to t = 6 :-



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 between 0 & 1, 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 between 0 & 1).

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 between 0 & 1, 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 between 0 & 1).

Etc.

^ NAND-Gate D Flip-Flop

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

Ben uses the RESISTOR-CAPACITOR method for EDGE-TRIGGERING.
However, the D flip-flops of the ICs that he later uses for the REGISTERS 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.

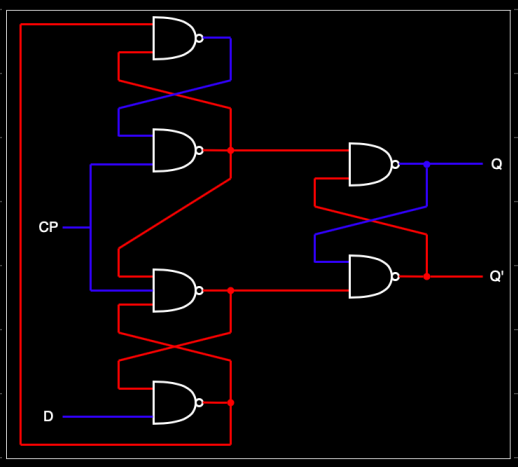
In real-world circuits, the duration of a clock cycle is MUCH MORE than this.

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]

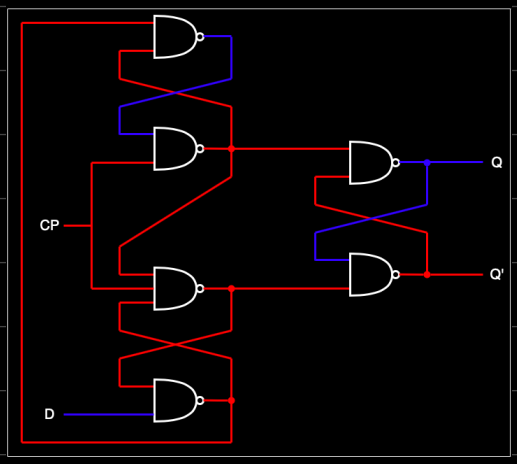
Let the initial STABLE state be CP = 0, D = 0, Q = 0 and Q' = 1 (CLEAR), and let the NEXT bit to be stored be 0 (which is present at the D input at t = 0 and whose SETUP time is completed), then 1, then 1, and then 0.

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).

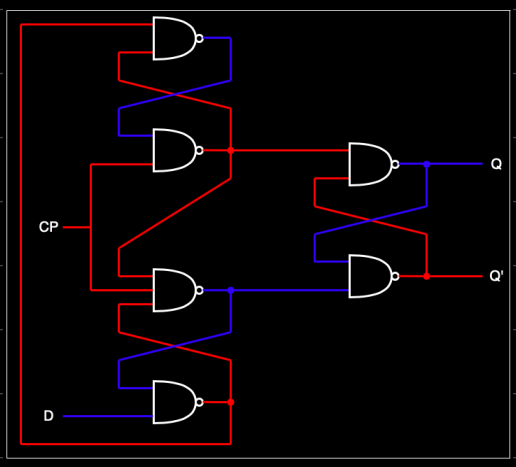
Till t = 0 :-



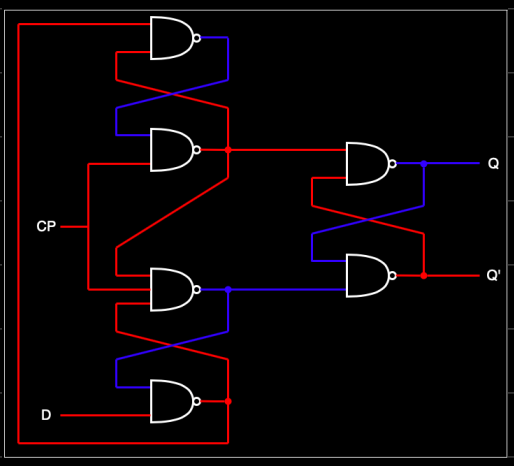
From t = 0 to t = 1 :-



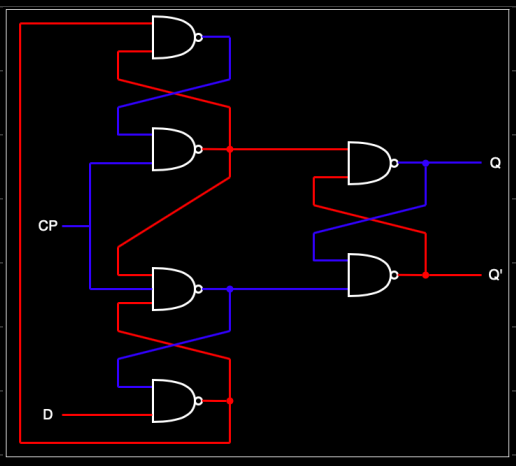
From t = 1 to t = 7 :-



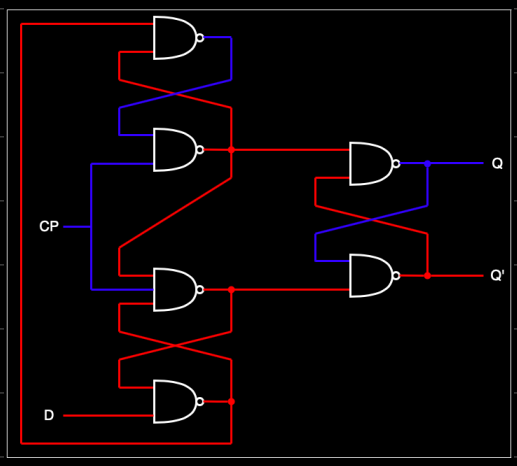
From t = 7 to t = 10 :-



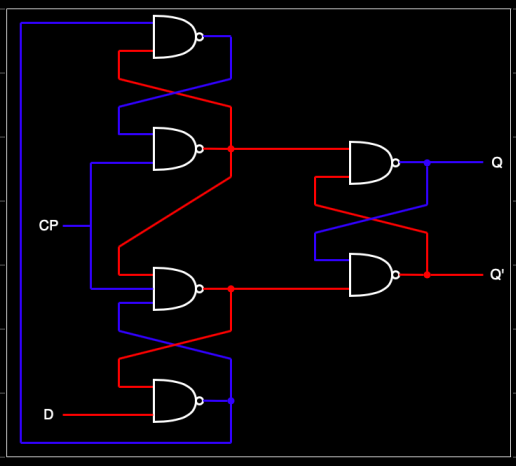
From t = 10 to t = 11 :-



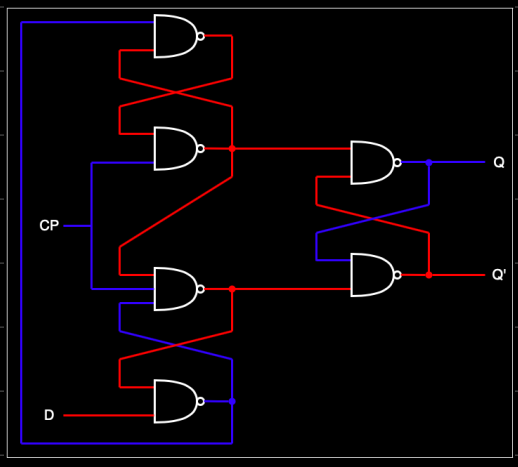
From t = 11 to t = 12 :-



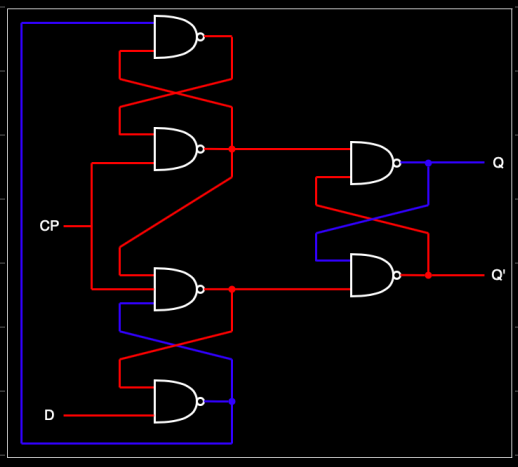
From t = 12 to t = 13 :-



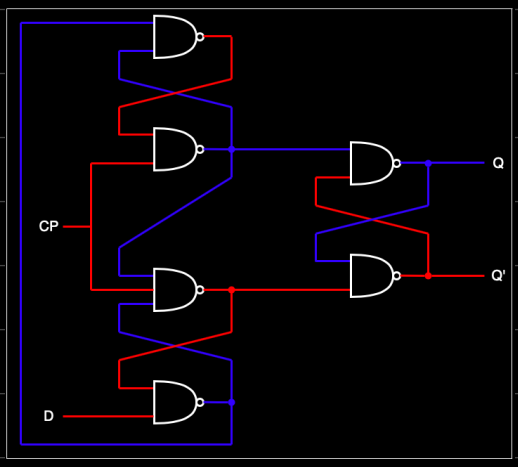
From t = 13 to t = 20 :-



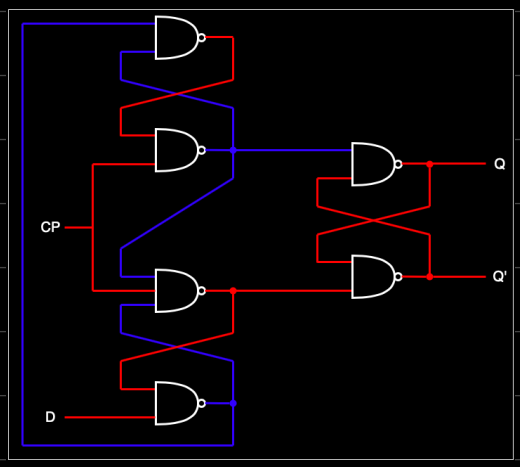
From t = 20 to t = 21 :-



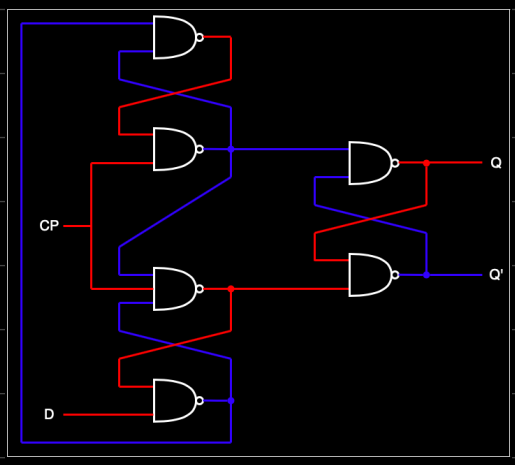
From t = 21 to t = 22 :-



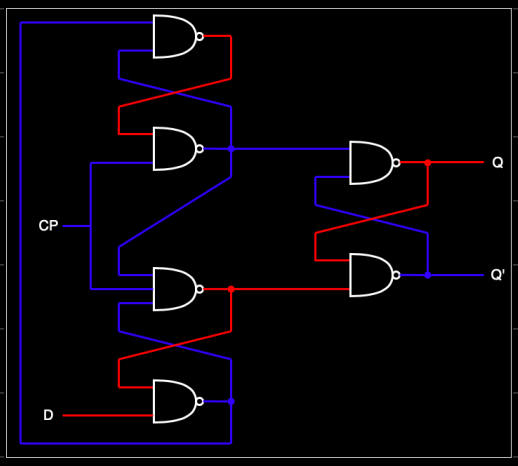
From t = 22 to t = 23 :-



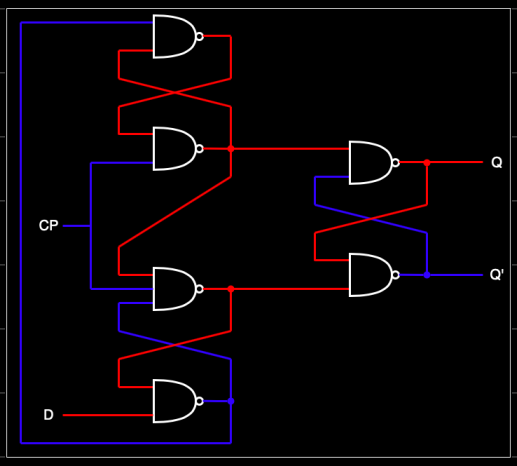
From t = 23 to t = 30 :-



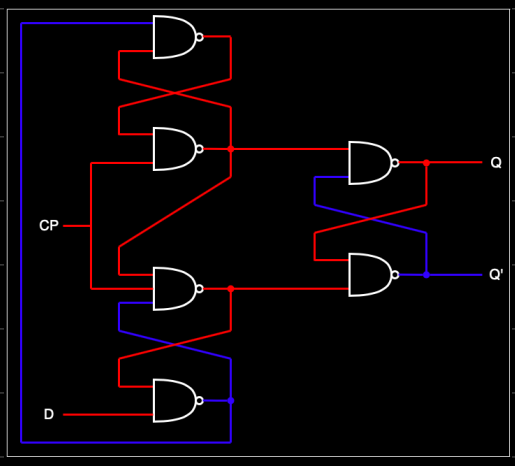
From t = 30 to t = 31 :-



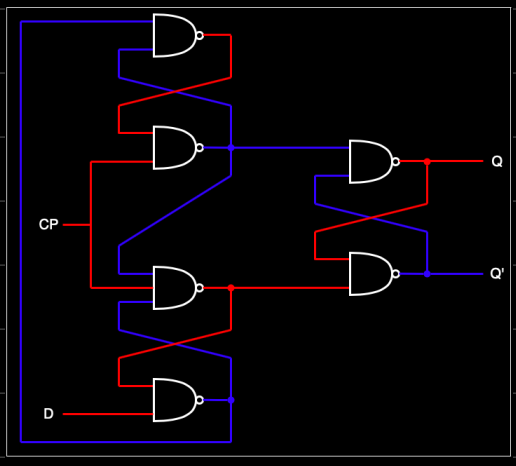
From t = 31 to t = 40 :-



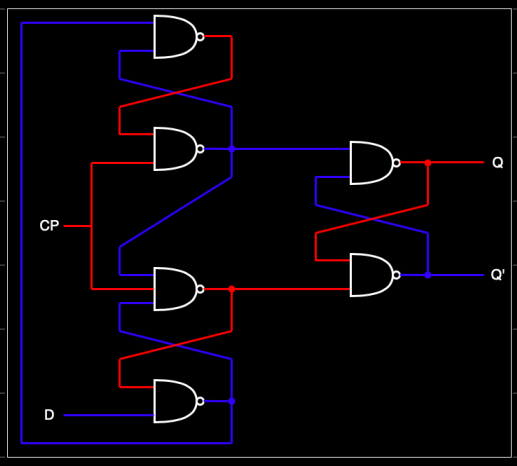
From t = 40 to t = 41 :-



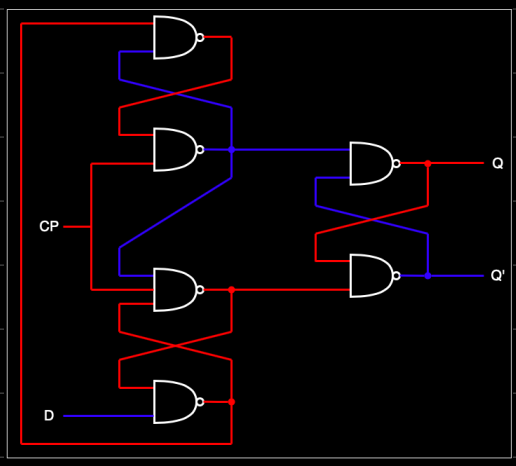
From t = 41 to t = 47 :-



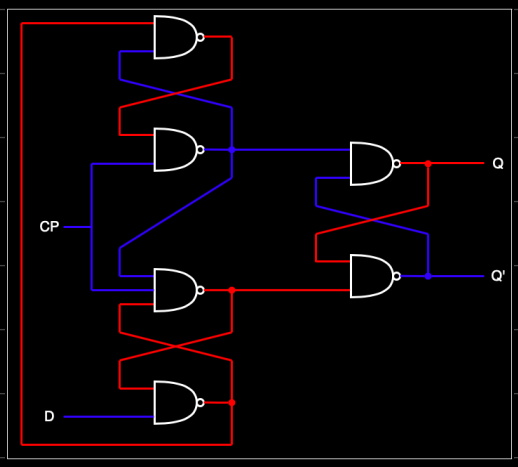
From t = 47 to t = 48 :-



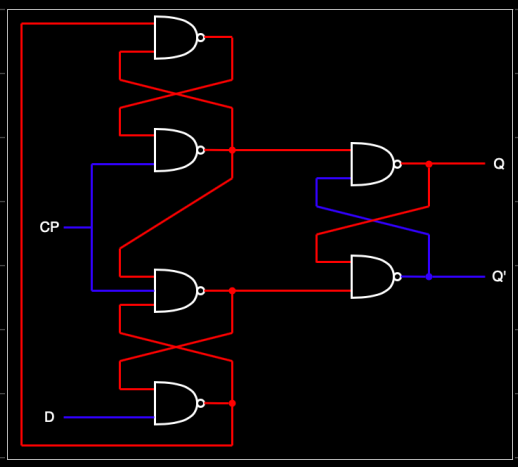
From t = 48 to t = 50 :-



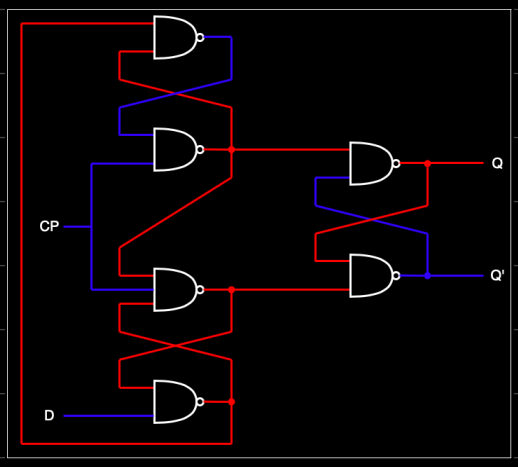
From t = 50 to t = 51 :-



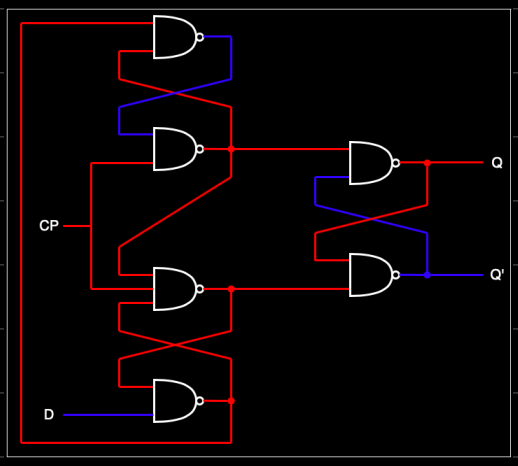
From t = 51 to t = 52 :-



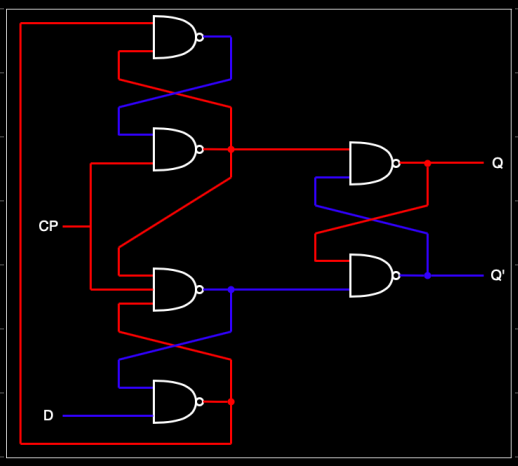
From t = 52 to t = 60 :-



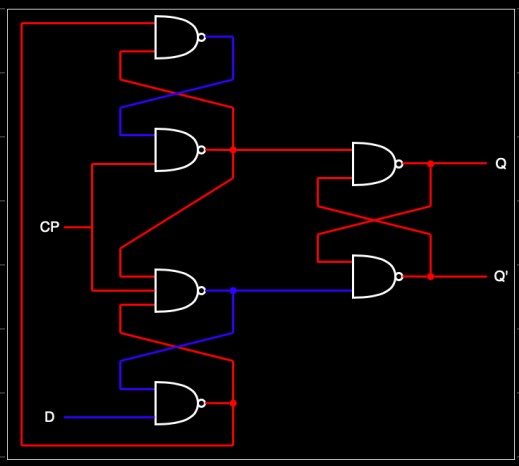
From t = 60 to t = 61 :-



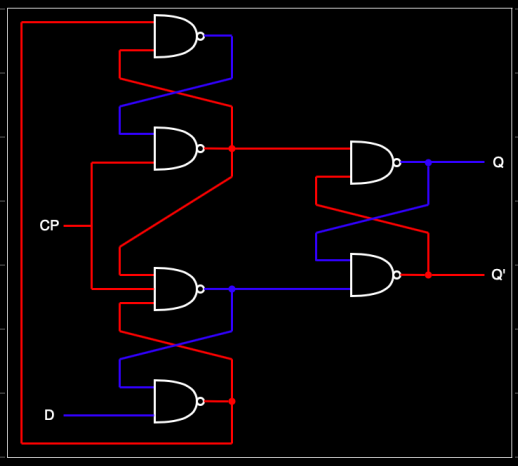
From t = 61 to t = 62 :-



From t = 62 to t = 63 :-



From t = 63 onwards :-

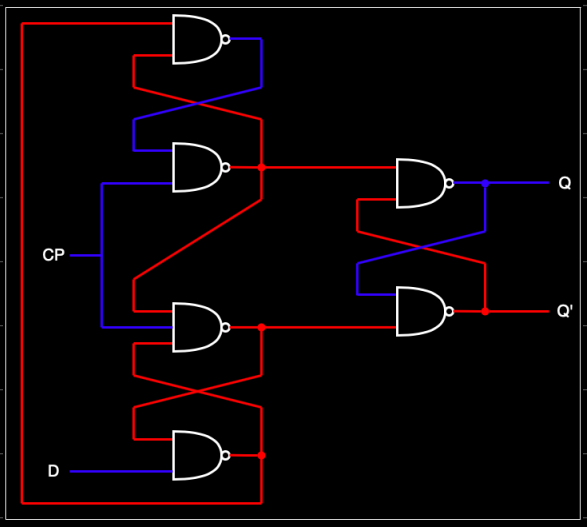


Explanation for the SETUP time and the HOLD time when the flip-flop is getting changed from CLEAR to SET :-

Let the initial STABLE state be

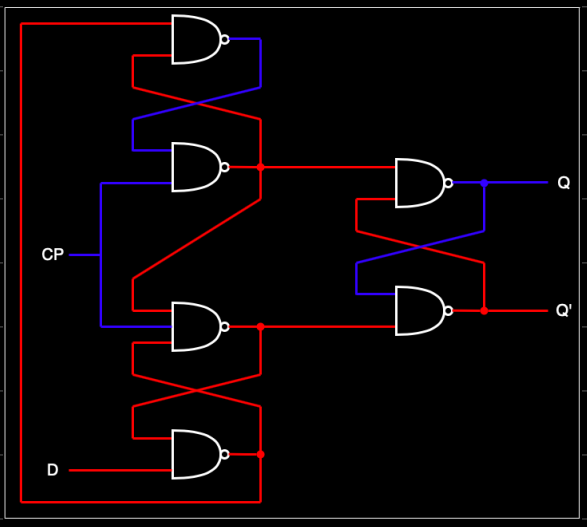
CP = 0, D = 0, Q = 0 and

Q' = 1 (CLEAR).

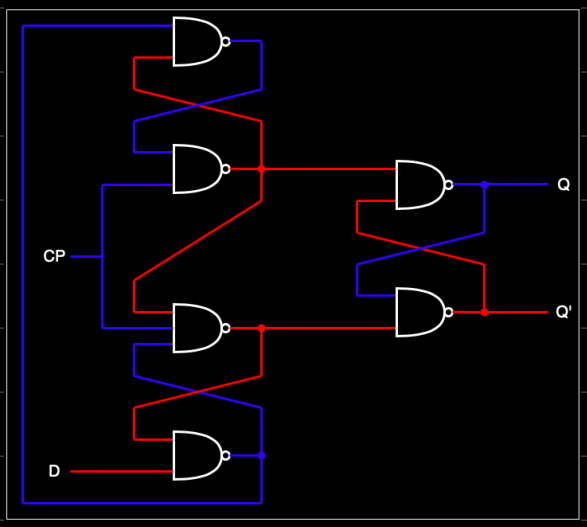


Let D get changed to 1 at t = 0.

From t = 0 to t = 1 –

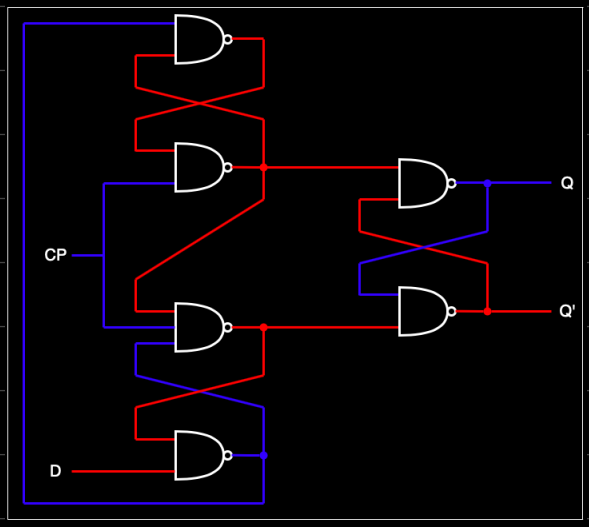


From t = 1 to t = 2 –



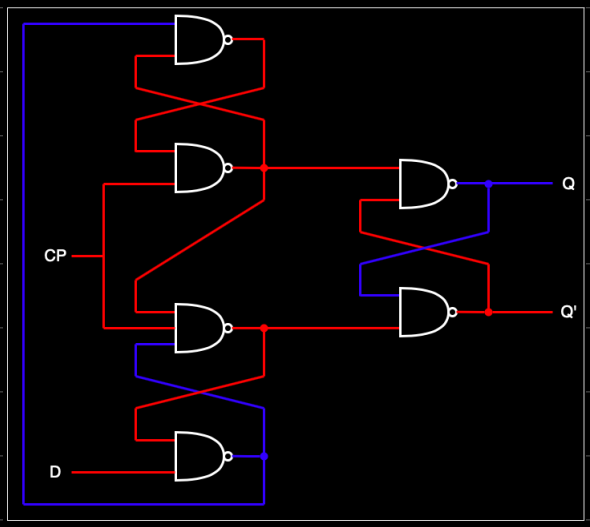
From t = 2 onwards –

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



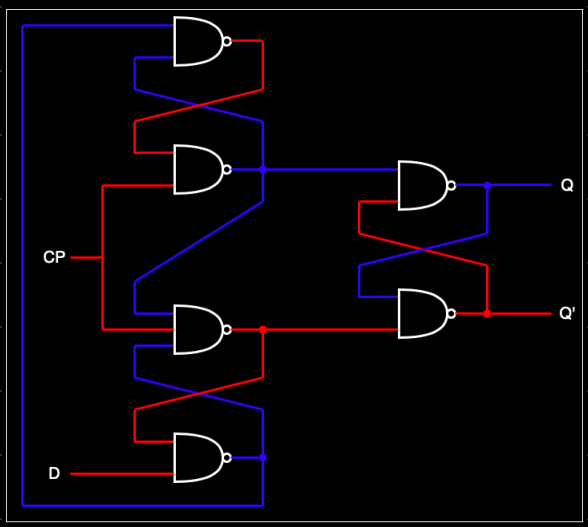
Now, let the initial STABLE state be CP = 0, D = 1, Q = 0 and Q' = 1 (CLEAR) and let CP get changed to 1 at t = 0.

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.



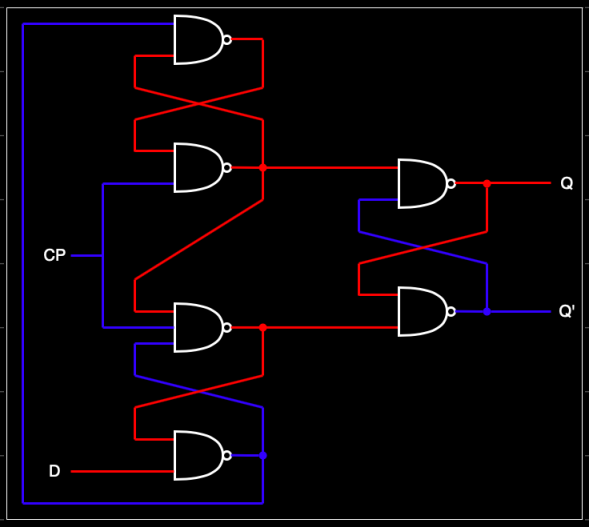
Therefore, the HOLD time is equal to the propagation delay through the UPPER gate to which CP is connected.

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

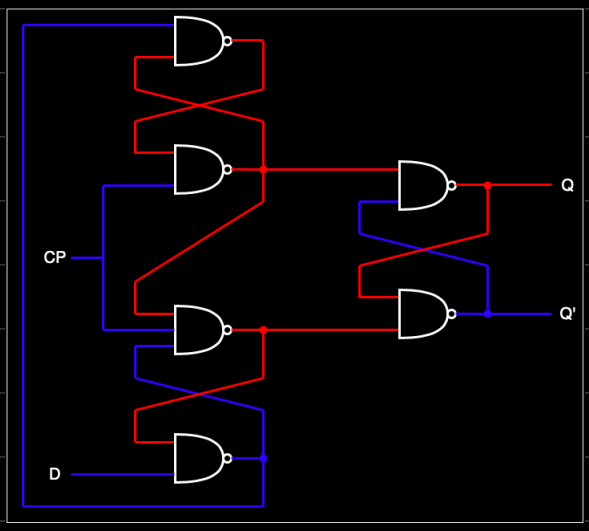
CP = 0, D = 1, Q = 1 and

Q' = 0 (SET).

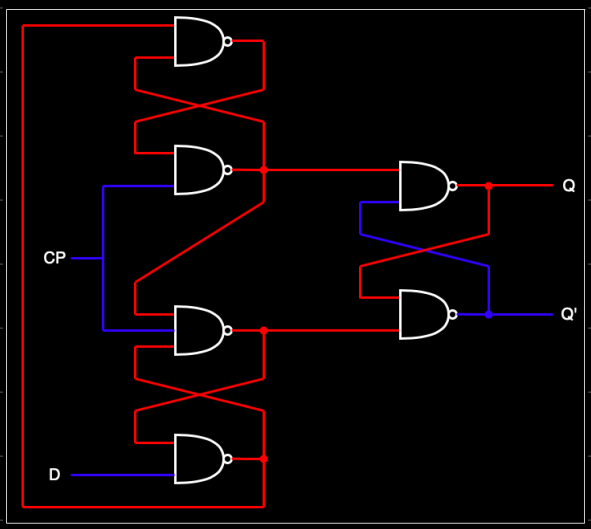


Let D get changed to 0 at t = 0.

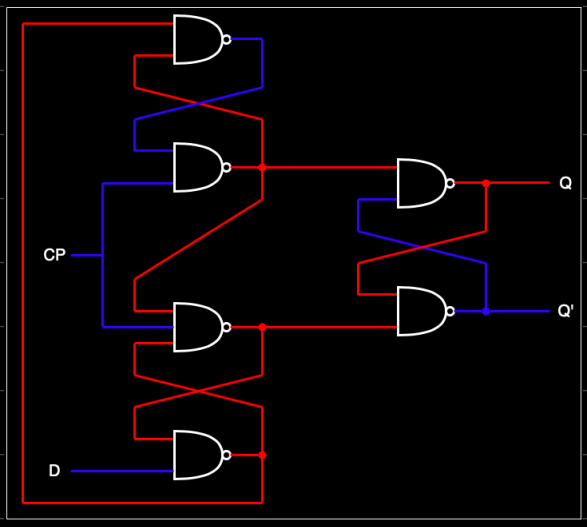
From t = 0 to t = 1 –



From t = 1 to t = 2 –

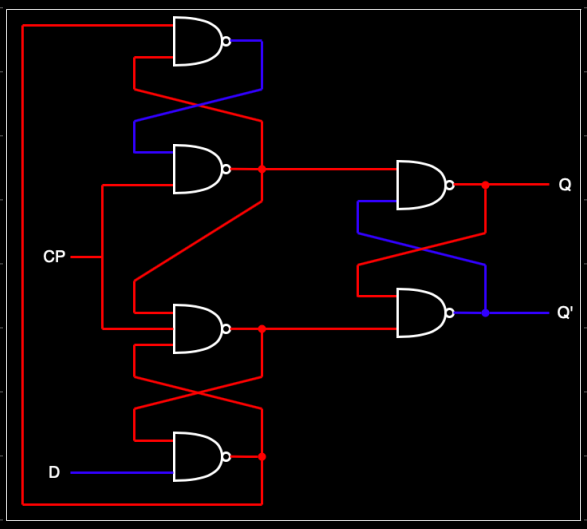


From t = 2 onwards –



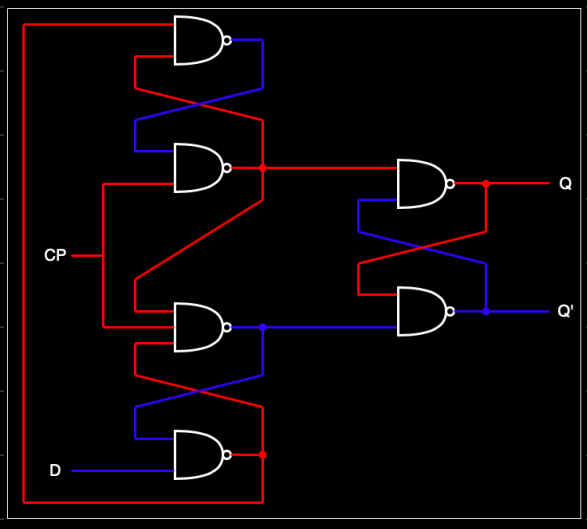
So, the SETUP time is equal to the propagation delays through the BOTTOMMOST gate and then through the TOPMOST gate. In the aforementioned transitions, from t = 50 onwards, it may SEEM like the SETUP time is 1 unit, but it should be noted that the FIRST part of SETUP, i.e. the propagation through the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

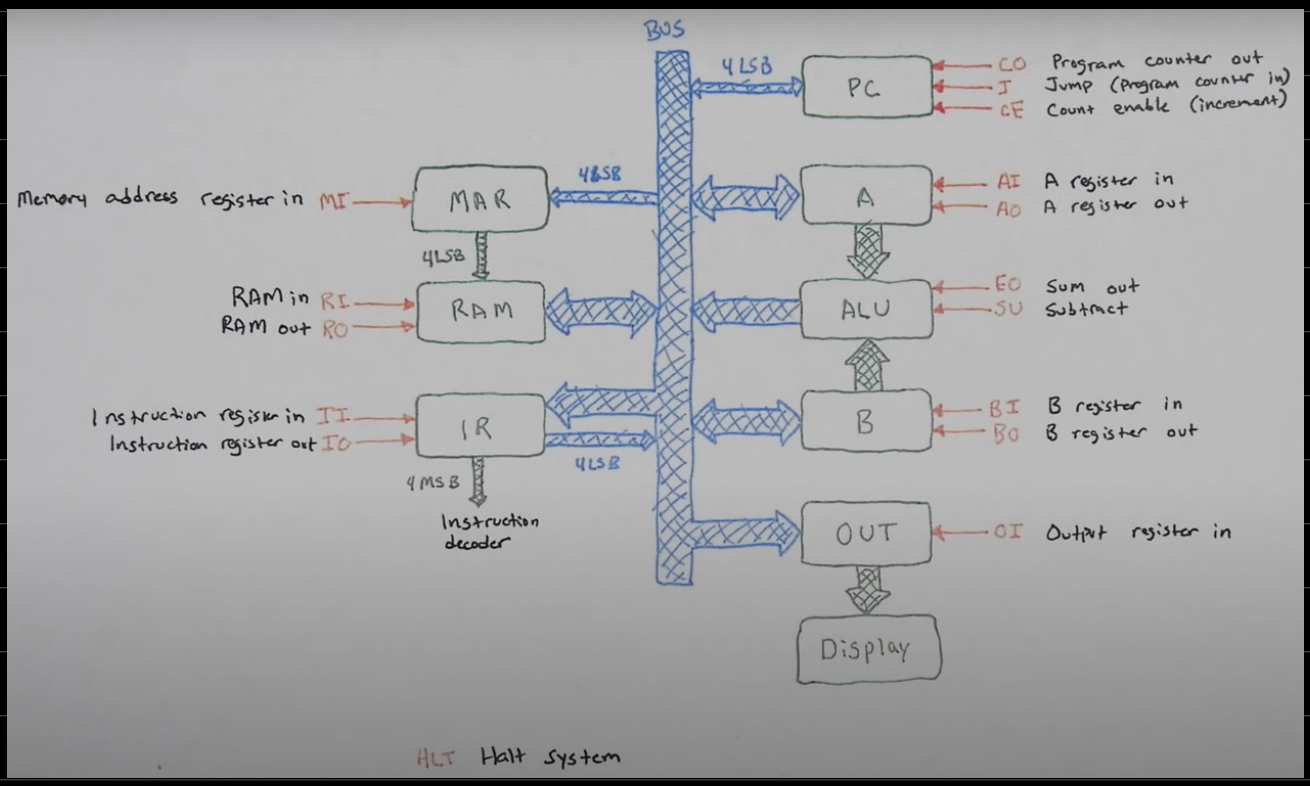
What happens if CP gets changed to 1 before the setup time is completed?

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.

^ Overview



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 (Flags Register In) is used.

For TTL ICs (such as the 74LS series), those INPUTS which are FLOATING (at 0, 1 or somewhere between 0 & 1, or oscillating among 0, 1 and somewhere between 0 & 1) (for eg., due to being DISCONNECTED, etc.) IDEALLY get converted to 1's.

But, in REAL-WORLD circuits, such inputs are EXTREMELY sensitive to various factors such as the EXTERNAL electrical NOISE, etc.

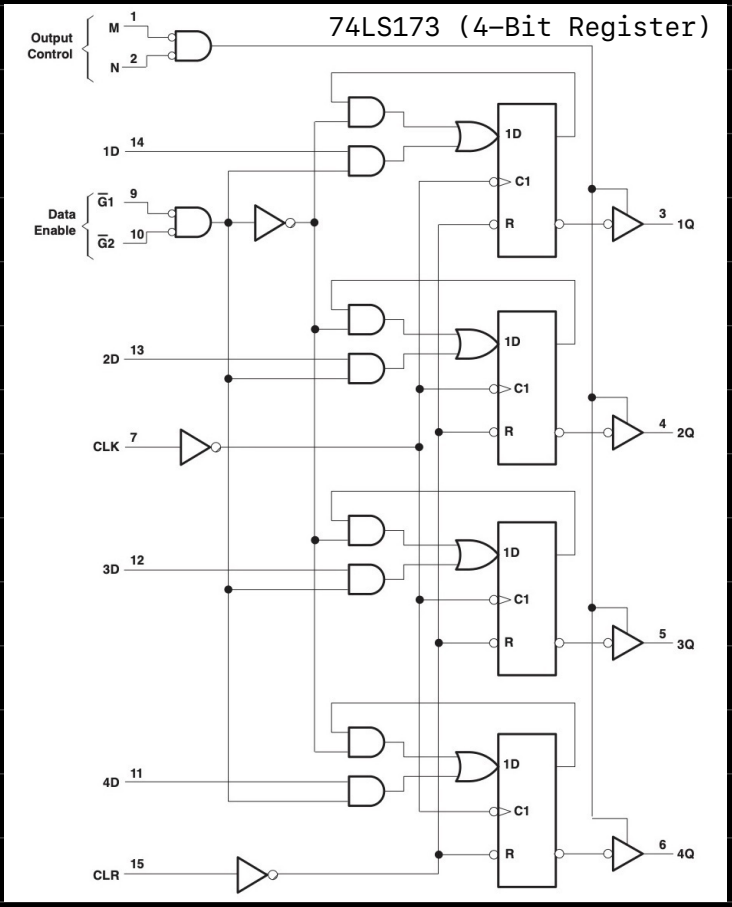
So, in REAL-WORLD circuits, such inputs may remain FLOATING (at 0, 1 or somewhere between 0 & 1, or oscillating among 0, 1 and somewhere between 0 & 1).

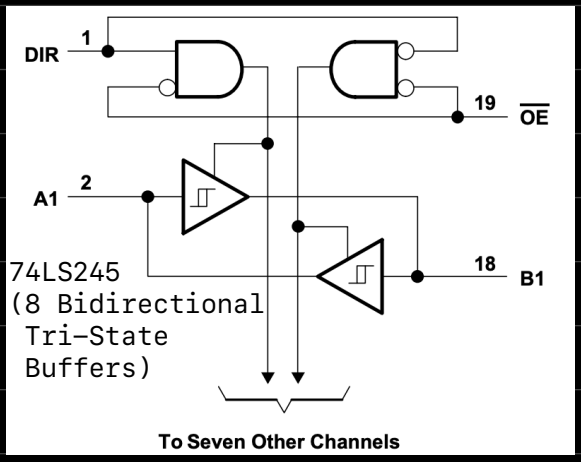
If the INPUT of ONE tri-state buffer whose OUTPUT is connected to a BUS wire is 0/1 (including FLOATING at 0/1), then the value on that BUS wire will become 0/1, provided that ALL OTHER tri-state buffers whose OUTPUTS are connected to that BUS wire are either DISABLED, or ENABLED with their INPUTS FLOATING somewhere between 0 & 1.

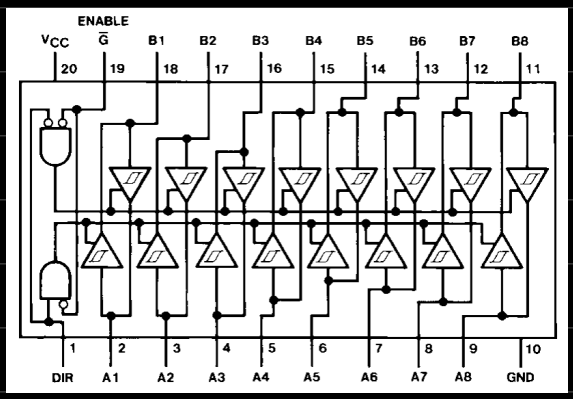
The BUS is constructed such that the values on its wires become 0's when the BUS is NOT being driven. So, if ALL tri-state buffers whose OUTPUTS are connected to a BUS wire are either DISABLED, or are ENABLED with their INPUTS FLOATING somewhere between 0 & 1, then the value on that BUS wire will become 0.

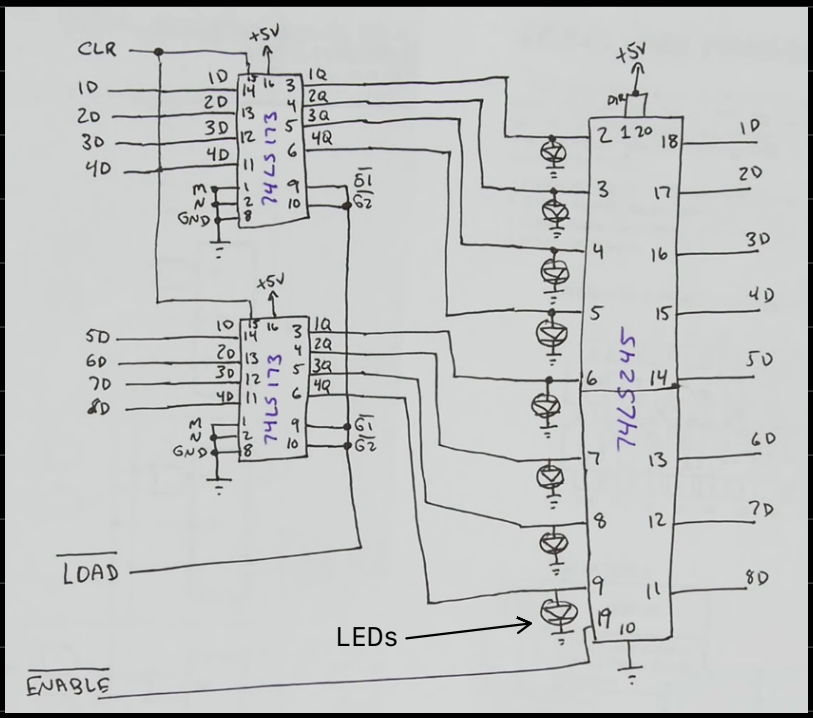
However, for the MOST part, there would be NO problems EVEN IF the values on the BUS wires would be FLOATING (at 0, 1 or somewhere between 0 & 1, or oscillating among 0, 1 and somewhere between 0 & 1) when the BUS is NOT being driven.
This is because, for eg., if ONE input of a NOR gate is 1 (including FLOATING at 1), then after its PROPAGATION delay, its OUTPUT will become 0 IRRESPECTIVE of the OTHER input (even if that OTHER input is FLOATING (at 0, 1 or somewhere between 0 & 1, or oscillating among 0, 1 and somewhere between 0 & 1)).
Similarly, the CLOCK input or the IN signal of a component being at 0 would determine the OUTPUT(s) of the corresponding logic gate(s) IRRESPECTIVE of the input(s) coming from the BUS (even FLOATING (at 0, 1 or somewhere between 0 & 1, or oscillating among 0, 1 and somewhere between 0 & 1).

^ A Register, B Register, Instruction Register









The M and N inputs of every 74LS173 are ALWAYS at 0, making the outputs of every 74LS173 ALWAYS enabled.

The DIR input of every 74LS245 is ALWAYS at 1, fixing the DIRECTION to be from A to B.

The only DIFFERENCE between the INSTRUCTION REGISTER and the OTHER REGISTERS is that the 4 MOST SIGNIFICANT output wires from the INSTRUCTION REGISTER's 74LS173 are NOT connected to its 74LS245.
So, these DISCONNECTED inputs of its 74LS245 MAY remain FLOATING somewhere between 0 & 1, causing the corresponding outputs to remain FLOATING when its 74LS245 is ENABLED.

In this case, there MAY be a problem if the values on the corresponding wires of the BUS would be FLOATING when the BUS is NOT being driven. This is because for the LDI (Load Immediate) instruction, the A REGISTER would be LOADING the values present on the BUS coming from the INTSRUCTION REGISTER, INCLUDING the 4 MOST SIGNIFICANT bits. This problem MAY occur EVEN THOUGH the BUS is constructed such that the values on its wires become 0's when the BUS is NOT being driven. This is because DURING the LDI instruction's execution, every DISCONNECTED input of the INSTRUCTION REGISTER's 74LS245 FLOATING at 1 would cause the value on the corresponding BUS wire to become 1. So, the ONLY solution to this problem is to FIX the inputs of the 4 MOST SIGNIFICANT tri-state buffers of its 74LS245 to 0. Doing so will ALSO ensure that 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.

In REAL-WORLD circuits, for ALL ICs, ALL unused INPUTS should be FIXED at 0 or 1, even for unused LOGIC GATES, and ALL unused OUTPUTS should remain DISCONNECTED.

^

So, AFTER the computer is powered up and BEFORE the RESET button is pressed, the OUTPUT of every GATED LATCH / FLIP-FLOP will be ARBITRARY (i.e. a STABLE 0, a STABLE 1 or OSCILLATING between 0 & 1, but NOT FLOATING) which MAY cause arbitrary CONTROL WORDS to get generated one after another.

However, the CLOCK inputs of ALL the components (EXCEPT the STEP COUNTER) will stay DISABLED during this time, preventing any IN operations from taking place.

To be answered later (?) –

What PROBLEMS, if any, will occur if the BUS is NOT constructed such that the values on its wires become STABLE 0's when the bus is NOT being driven?

Also, what PROBLEMS, if any, will occur if the constituent LOGIC GATES of the ICs that Ben uses are NOT constructed to convert DISCONNECTED inputs and FLOATING inputs to 1?

The ICs that Ben uses for the REGISTERS consist of D FLIP-FLOPS (or equivalent), whereas the IC that he uses for the RAM consists of GATED D LATCHES (or equivalent).

So, for the RAM, EDGE-TRIGGERING is achieved using the RESISTOR-CAPACITOR method.

The ICs that he uses for the RAM and for the OUTPUT REGISTER don't have built-in LOAD / WRITE ENABLE / IN signals, whereas the ICs that he uses for ALL OTHER REGISTERS have built-in LOAD / WRITE ENABLE / IN signals. So, for the RAM and for the OUTPUT REGISTER, he ANDs the CLOCK input with external LOAD / WRITE ENABLE / IN signals.

AFTER the computer is powered up, the RESET button is pressed, which clears (STABLE 0) ALL the components, EXCEPT the RAM.

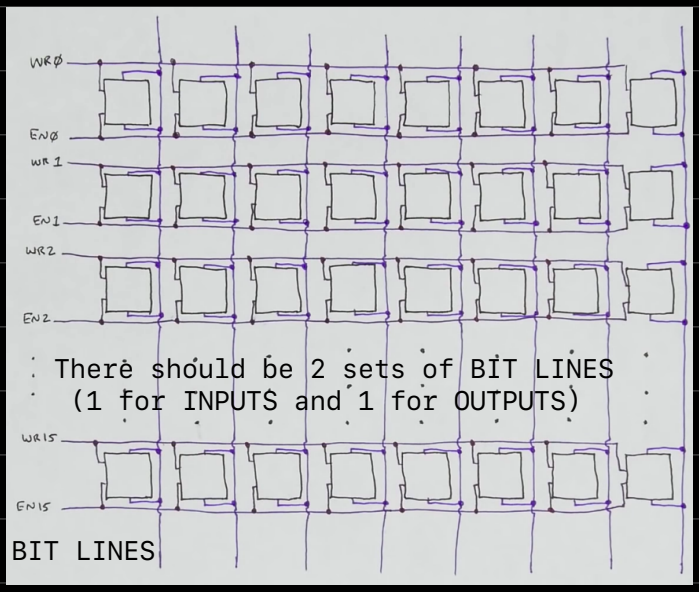
Hence, for every GATED D LATCH of the RAM (or equivalent), the output may be a STABLE 0, a STABLE 1 or OSCILLATING BETWEEN 0 & 1.

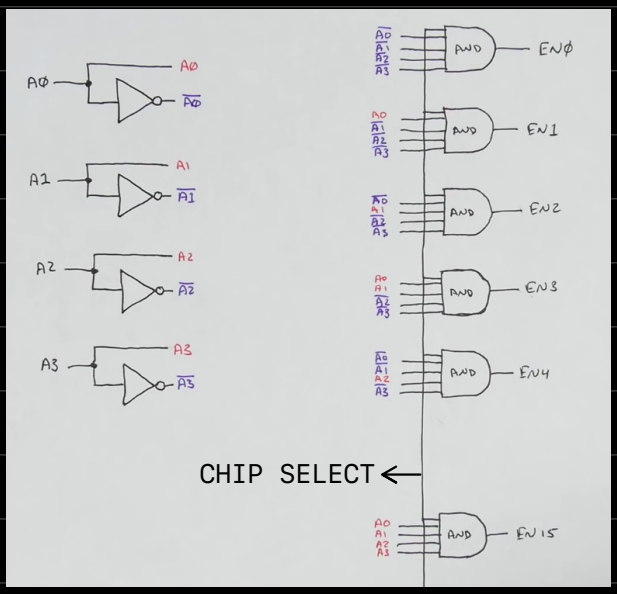
When required, the contents of the RAM will be written to. Doing so will NOT cause a problem because, for eg., for the NOR-gate SR latch, after S and R become CONSTANT at 1/0 and 0/1, respectively, then NO MATTER the current state of the latch, it will result in a STABLE SET/CLEAR state after a LONG ENOUGH time.

The time DURATION for which the RESET button is pressed is LONG ENOUGH for signals to PROPAGATE through the RESET circuit and then through the individual ICs, signals to PROPAGATE for the CONTROL WORD corresponding to the ADDRESS 0000 000 00 (i.e. CO = 1, MI = 1 & other signals = 0) to get generated, and signals to PROPAGATE after CO, MI & other signals become 1, 1 & 0, respectively.

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

AFTER the RESET button is released, the program to be run is MANUALLY loaded into the RAM using SWITCHES. A LONG ENOUGH time passes for signals to PROPAGATE BEFORE the PUSH BUTTON used for WRITING during MANUAL programming is pressed. Similarly, the time DURATION for which the PUSH BUTTON is pressed is LONG ENOUGH for signals to PROPAGATE.





The RAM's CHIP SELECT is always ENABLED, making the RAM's outputs always ENABLED. During NON-MANUAL mode, the MEMORY ADDRESS REGISTER's outputs determine the ADDRESS of the RAM, and the values on the BUS determine the DATA inputs of the RAM.

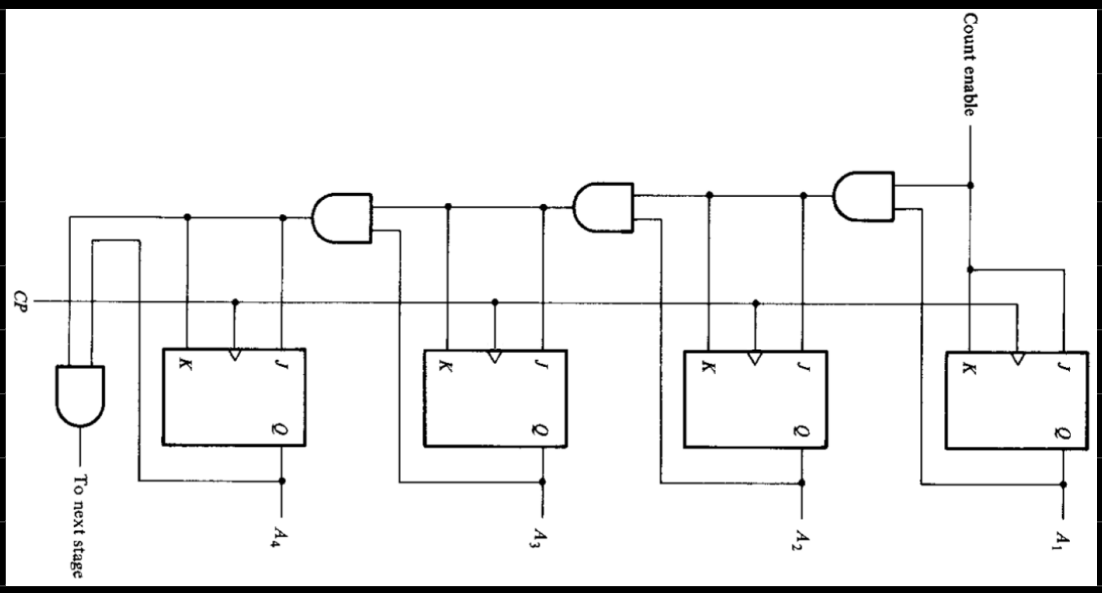
For reading from the RAM, internally, the CHIP SELECT input of the IC is ANDed with the outputs of the built-in ADDRESS DECODER's AND gates (or equivalent), and the outputs of these AND gates become the ENABLE inputs of the corresponding internal TRI-STATE BUFFERS (or equivalent) connecting the outputs of the corresponding GATED D LATCHES (or equivalent) to the first set of BIT LINES (i.e. the RAM'S internal BUS wire

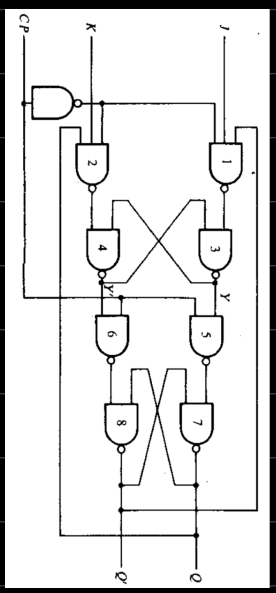
So, due to the CHIP SELECT always being ENABLED, whenever the ADDRESS of the RAM changes, the propagation delays of the RAM's built-in ADDRESS DECODER's INVERTERS (or something equivalent) MAY cause the CONTENTS at the OLD address and the NEW address to get output onto the BIT LINES SIMULTANEOUSLY. However, such a situation would last ONLY for a VERY SHORT amount of time, NOT causing any damage to the BIT LINES.
The RAM OUT signal becomes the ENABLE input of the external TRI-STATE BUFFERS' IC.

For writing to the RAM, internally, the second set of BIT LINES (or equivalent) is used to connect the DATA inputs to the inputs of the corresponding GATED D LATCHES (or equivalent). The RAM IN signal is ANDed with the CLOCK input (after using the RESISTOR-CAPACITOR method for EDGE-TRIGGERING), which becomes the WRITE ENABLE input be of the built-in ADDRESS DECODER's AND gates (or equivalent). The outputs of these AND gates become the ENABLE/CLOCK inputs of the corresponding GATED D LATCHES.

The EEPROMs' outputs are always ENABLED.
The INTSTRUCTION REGISTER's, the STEP COUNTER's and the FLAGS REGISTER's outputs determine the ADDRESS of each EEPROM of the CONTROL UNIT. So, whenever the ADDRESS of an EEPROM of the CONTROL UNIT changes, the propagation delays of the EEPROM's built-in ADDRESS DECODER's INVERTERS (or something equivalent) MAY cause the CONTROL WORDS at the OLD address and the NEW address to get output SIMULTANEOUSLY (via the EEPROM's built-in OR gates or something equivalent), POSSIBLY causing the OUT signals of MULTIPLE components to get enabled SIMULTANEOUSLY. Similarly, if a component's OUT signal is currently ENABLED, i.e. if it is currently outputting its contents onto the BUS, and if its OUT signal gets DISABLED according to the NEXT CONTROL WORD, then the p delays of that component's external TRI-STATE BUFFERS MAY cause that component to STOP outputting its contents onto the BUS AFTER some other component's OUT signal gets ENABLED according to that NEXT CONTROL WORD, POSSIBLY causing the two components to output their contents onto the BUS SIMULTANEOUSLY. However, such situations would last ONLY for VERY SHORT amounts of time, NOT causing any damage to the BUS.

The PROGRAM COUNTER and the 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 gated latch, and it goes through an internal INVERTER to the MASTER gated latch.





For the PROGRAM COUNTER, AFTER the computer is powered up and BEFORE the RESET button is pressed, the CLOCK inputs of all MASTER/SLAVE gated latches become 1/0 after signals get PROPAGATED through the internal INVERTERS.

AFTER it gets reset and its RESET input gets DISABLED, the CLOCK inputs of all MASTER/SLAVE gated latches stay at 1/0. But, as its COUNT ENABLE signal has already been at 0 since the time during which the RESET button was pressed, the outputs of all MASTER/SLAVE gated latches stay at 0.

The CLOCK input of the PROGRAM COUNTER goes through an external INVERTER and then becomes the CLOCK input of the STEP COUNTER.

So, for the STEP COUNTER, AFTER the computer is powered up and BEFORE the RESET button is pressed, the CLOCK inputs of all MASTER/SLAVE gated latches become 0/1 after signals get PROPAGATED through the external INVERTER and the internal INVERTERS.

AFTER it gets reset and its RESET input gets DISABLED, the CLOCK inputs of all MASTER/SLAVE gated latches stay at 0/1. So, even though its COUNT ENABLE signal is always at 1, the outputs of all MASTER/SLAVE gates latches stay at 0.

No need to specifically talk about the propagations through the internal AND gates, just like there's no need to specifically talk about the propagations through the internal gates of the flip-flops.