(*) Building an 8-bit Breadboard Computer

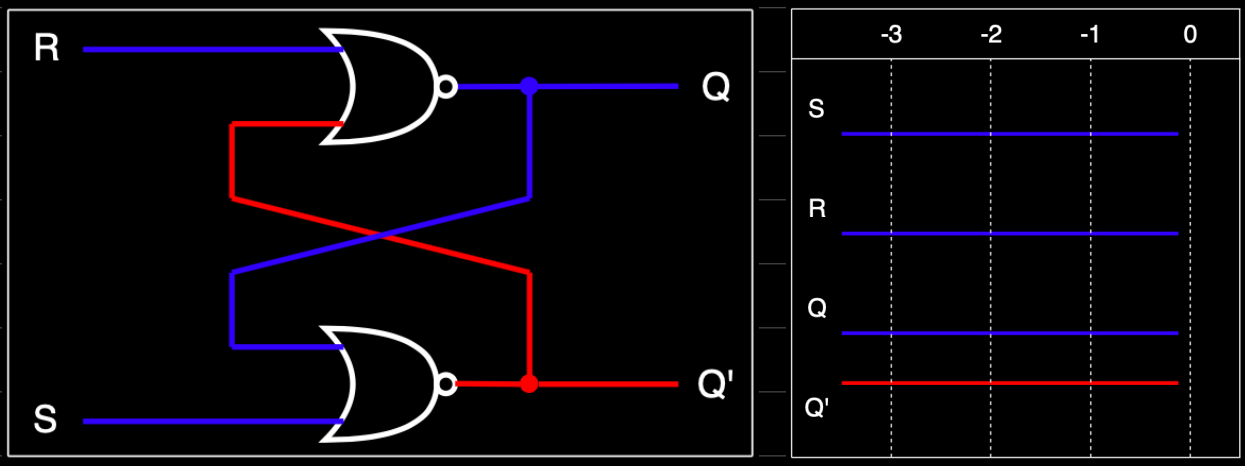
by Ben Eater

(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.

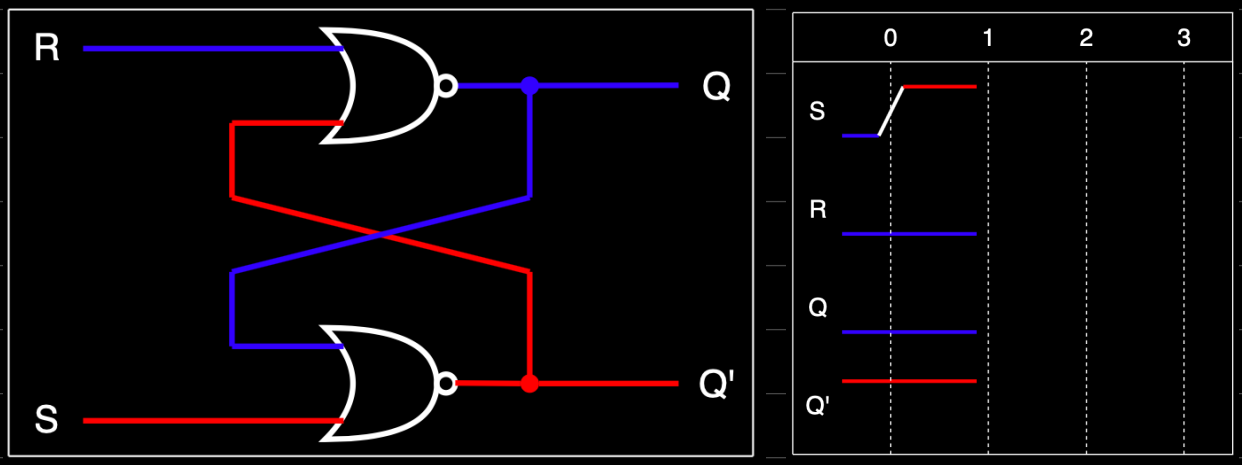
So, the EFFECTS of the changed inputs will be visible only AFTER the propagation delays.

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.

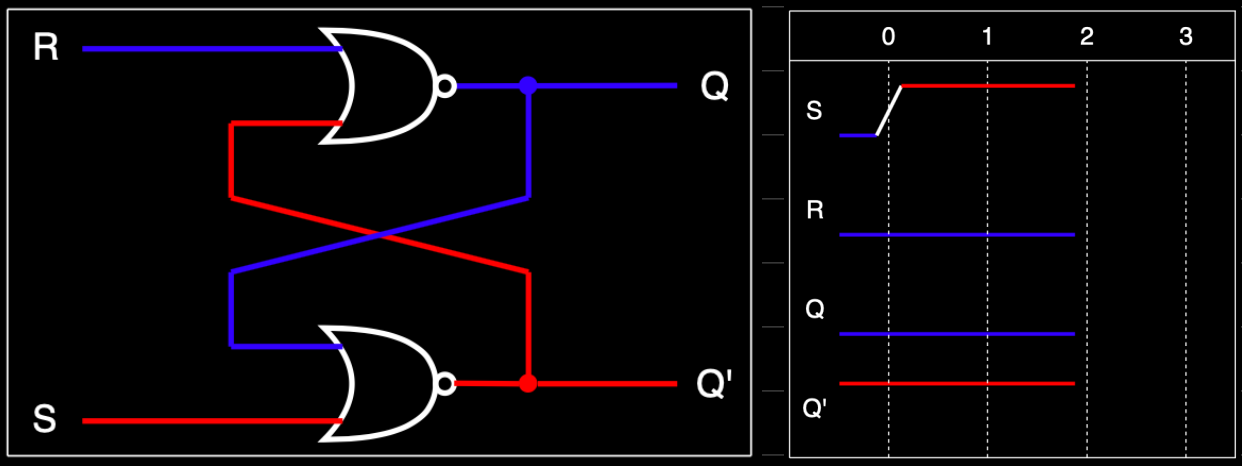
For eg., if the propagation delay of a logic gate is 3 units, then its OUTPUT at t = 3.5 units (say) depends upon its INPUTS at t = 0.5 units.

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

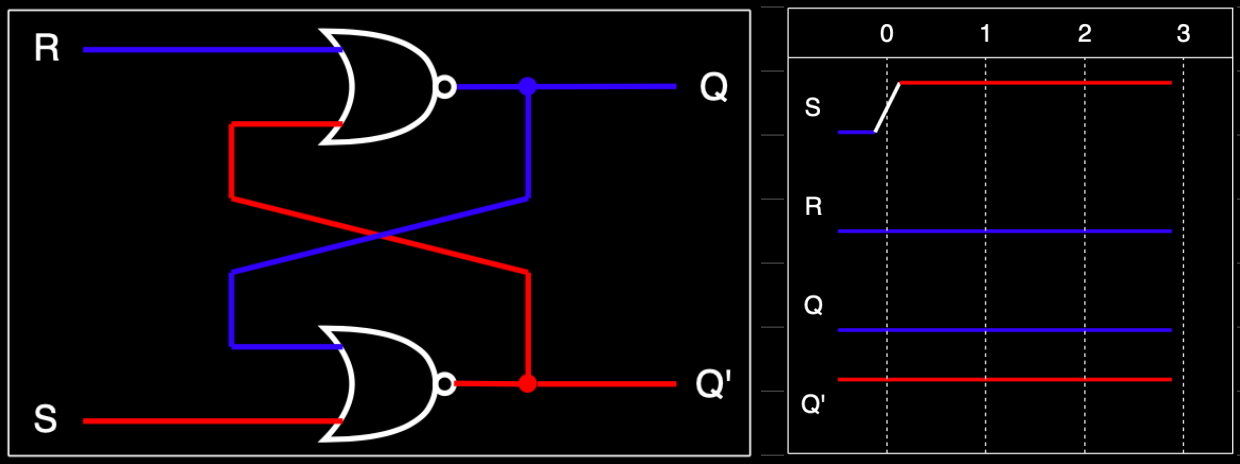
From t = 0 to t = 1 :-



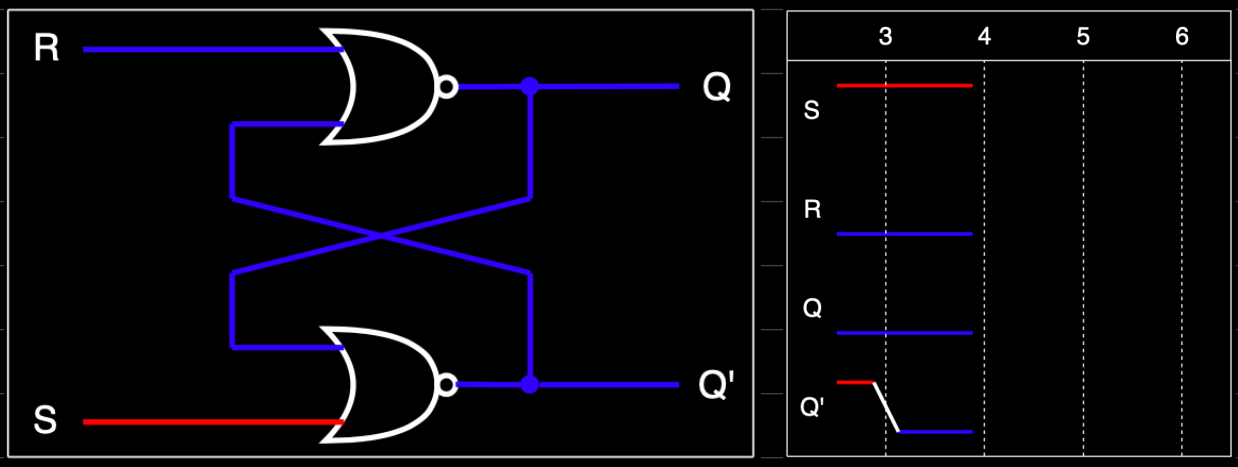
From t = 1 to t = 2 :-



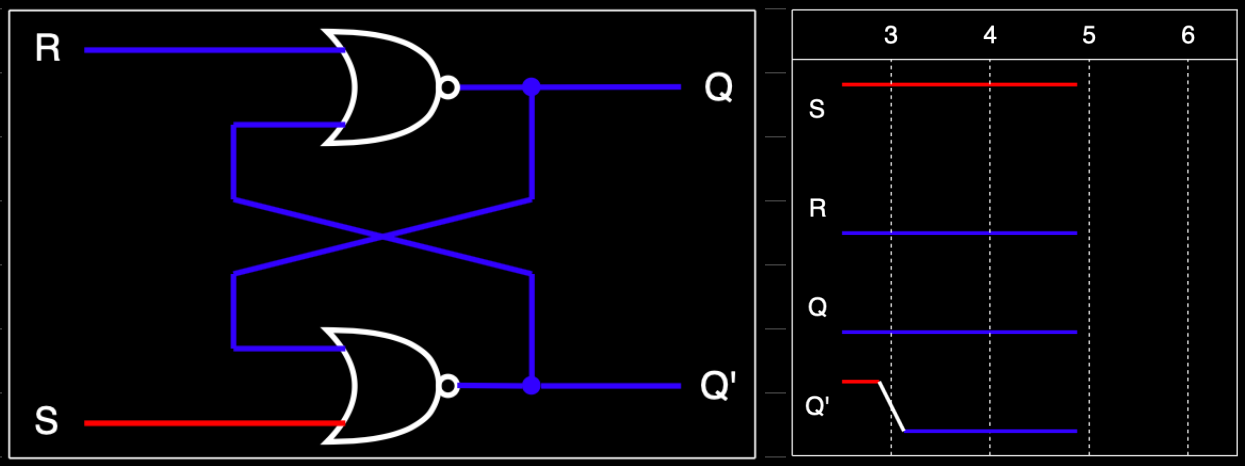
From t = 2 to t = 3 :-



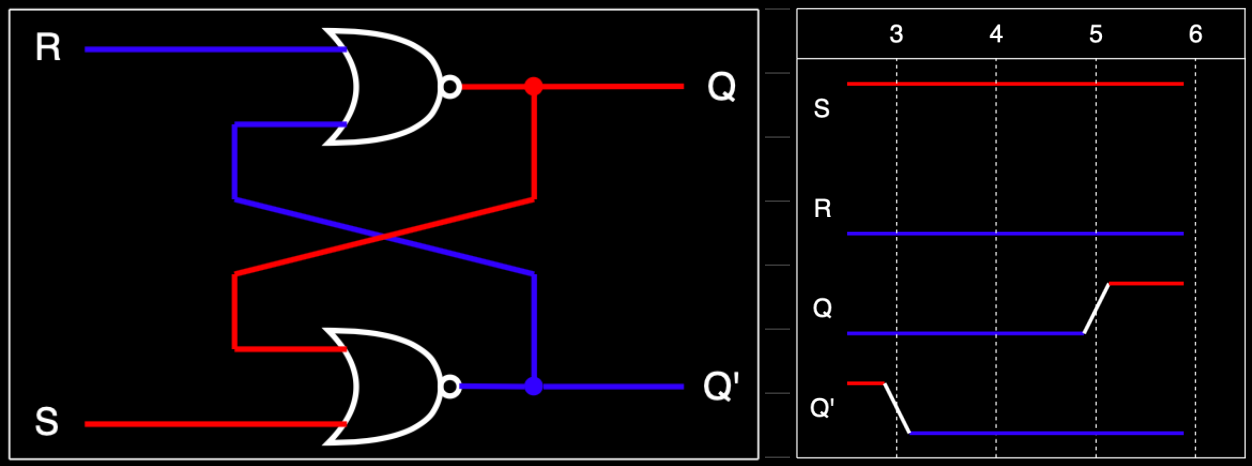
From t = 3 to t = 4 :-



From t = 4 to t = 5 :-



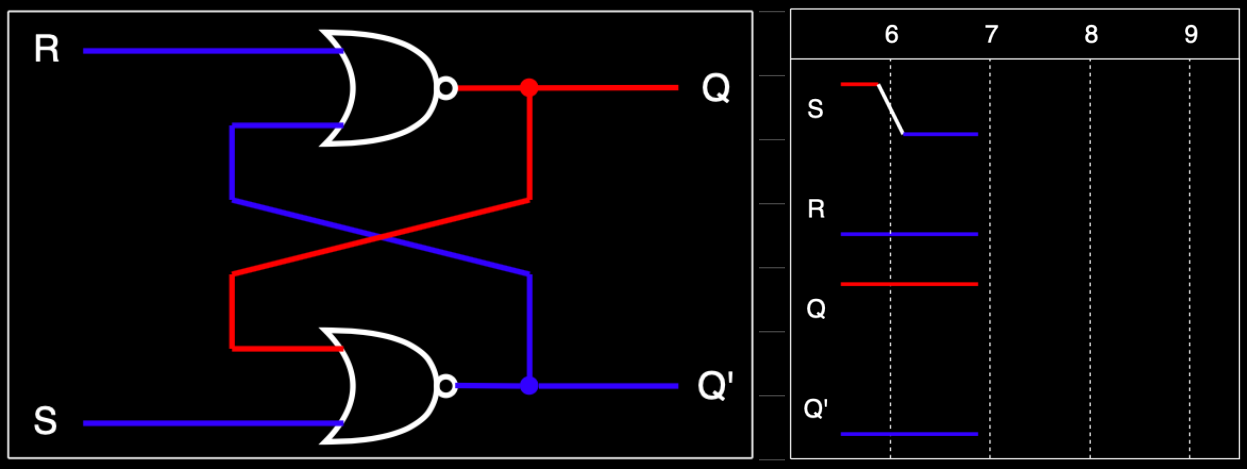
From t = 5 to t = 6 :-



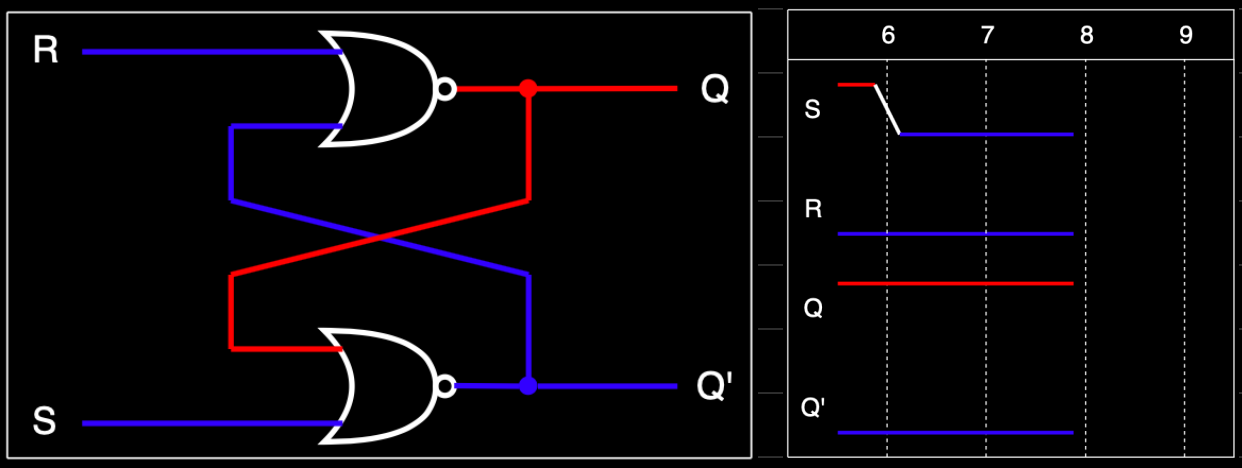
Now, the latch will stay STABLE in the SET state, even if S gets changed back to 0.

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

From t = 6 to t = 7 :-



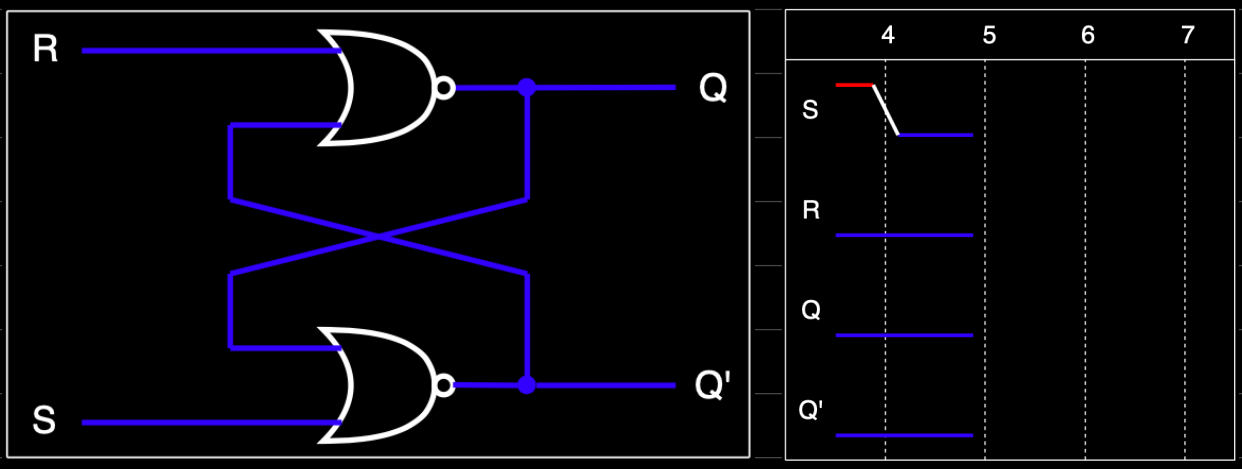
From t = 7 to t = 8 :-



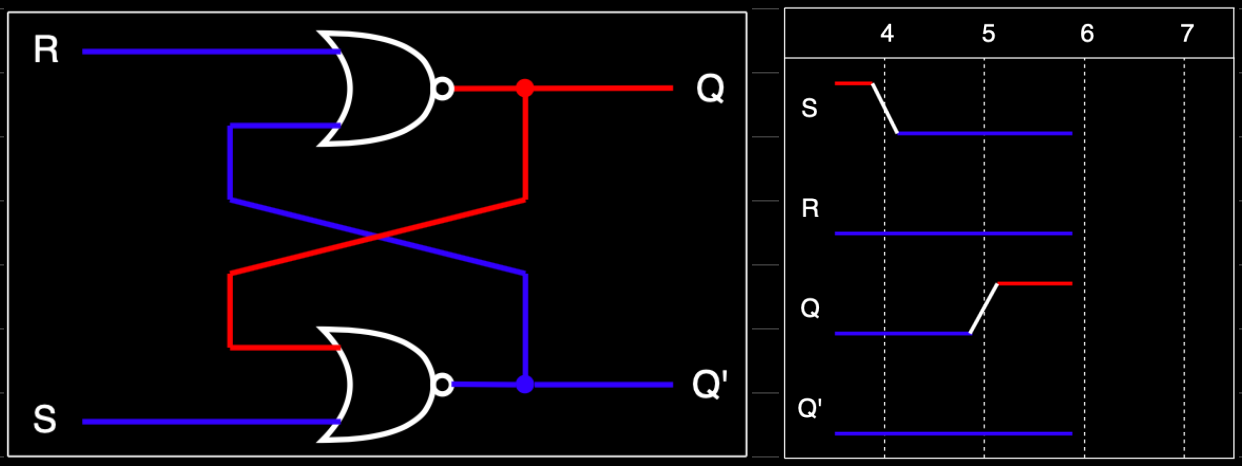
Hence, the latch will stay STABLE in the SET state, as long as S was maintained at 1 for a LONG ENOUGH time, i.e. if S gets changed back to 0 AFTER t = 5, then the latch will result in a STABLE SET state.

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

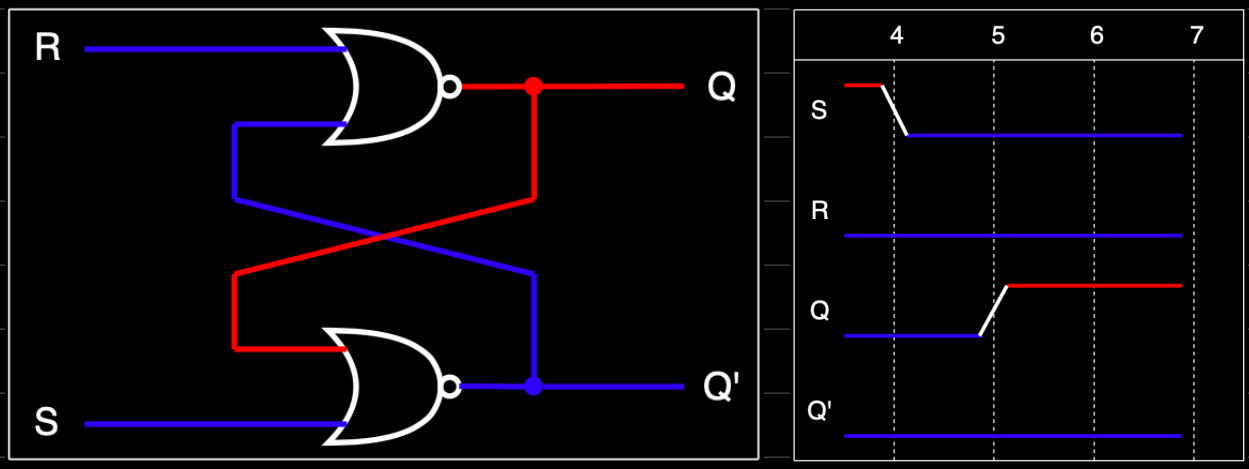
From t = 4 to t = 5 :-



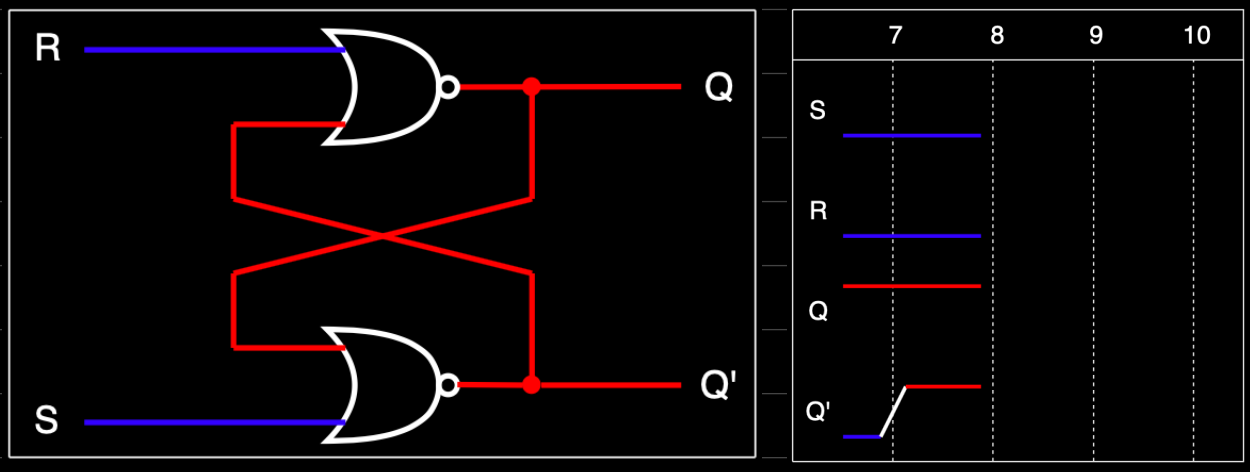
From t = 5 to t = 6 :-



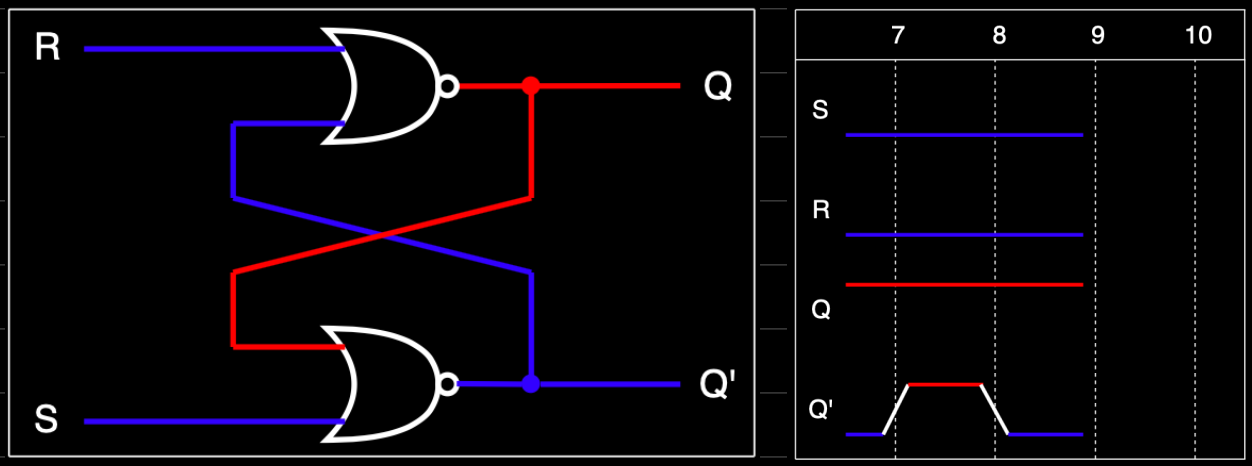
From t = 6 to t = 7 :-



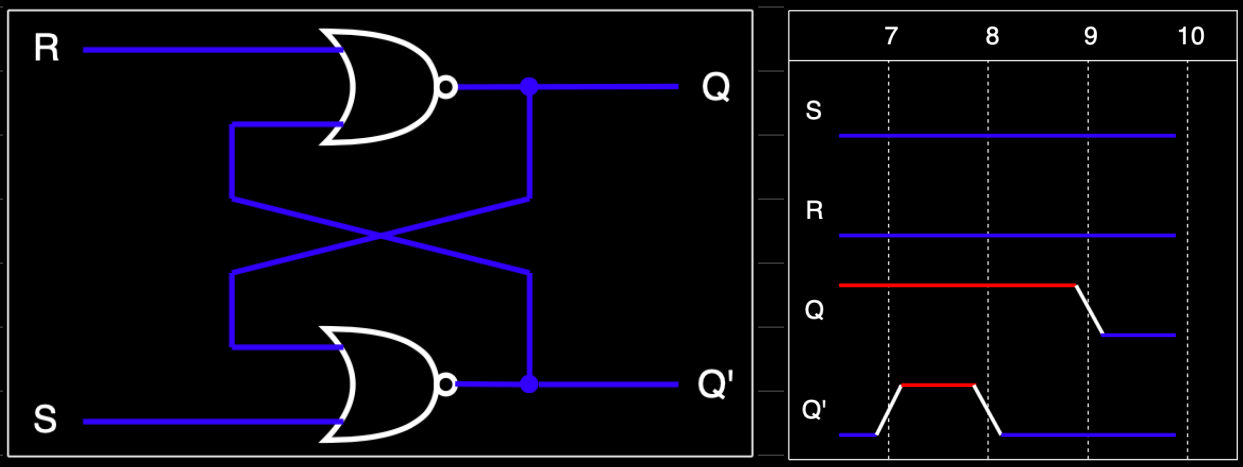
From t = 7 to t = 8 :-



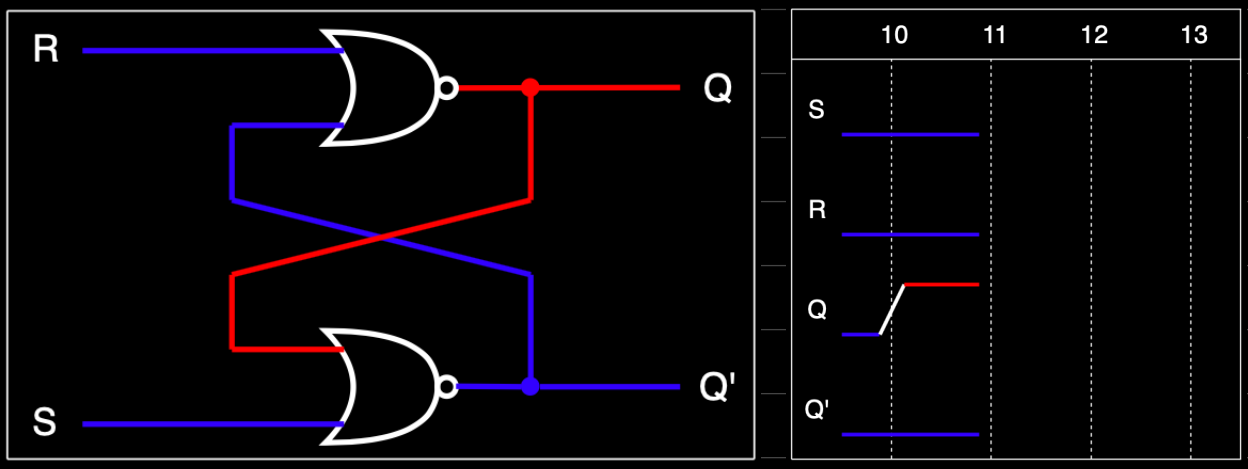
From t = 8 to t = 9 :-



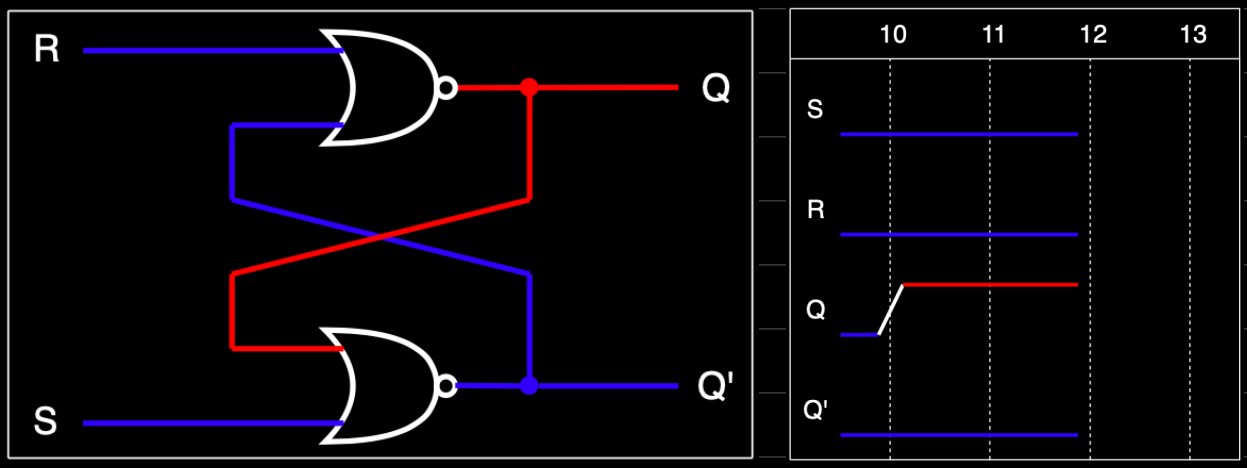
From t = 9 to t = 10 :-



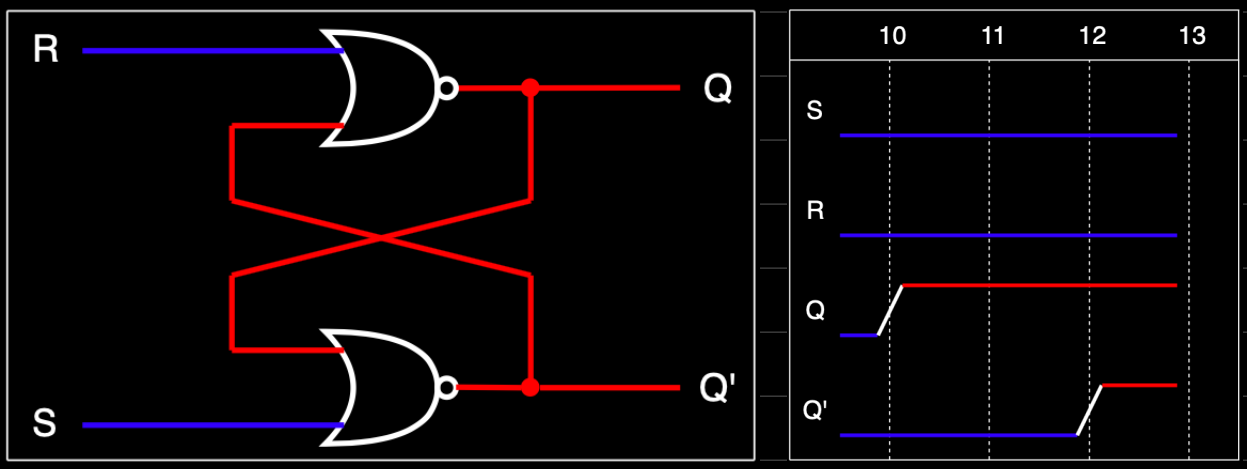
From t = 10 to t = 11 :-



From t = 11 to t = 12 :-



From t = 12 to t = 13 :-

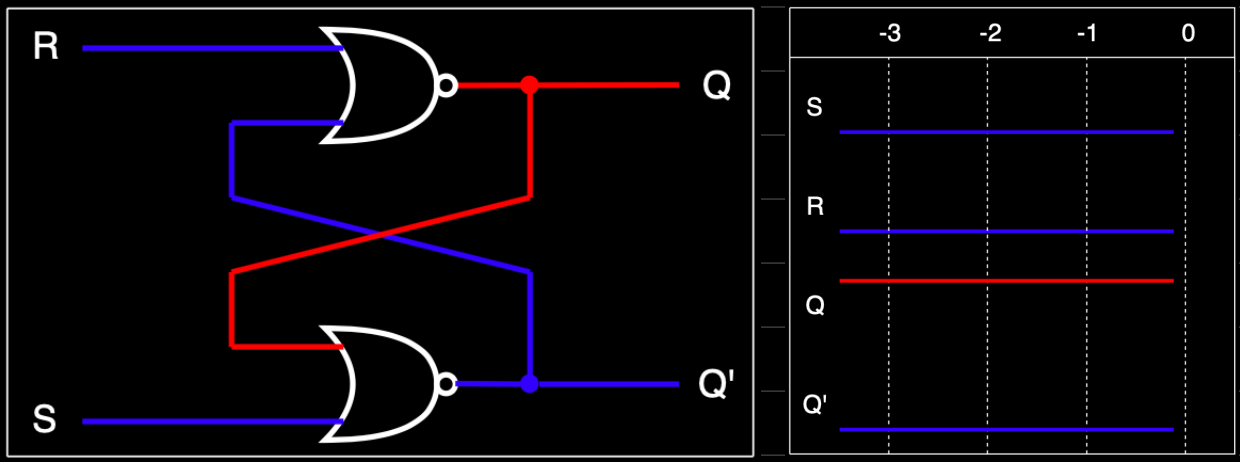


The latch will keep oscillating like this FOREVER. Previously, when the latch was being SET, it temporarily switched to an INVALID state, i.e. Q = Q' = 0, but then it became STABLE in the SET state. But, in this case, the latch will NEVER become STABLE in ANY state. Therefore, if S is NOT maintained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

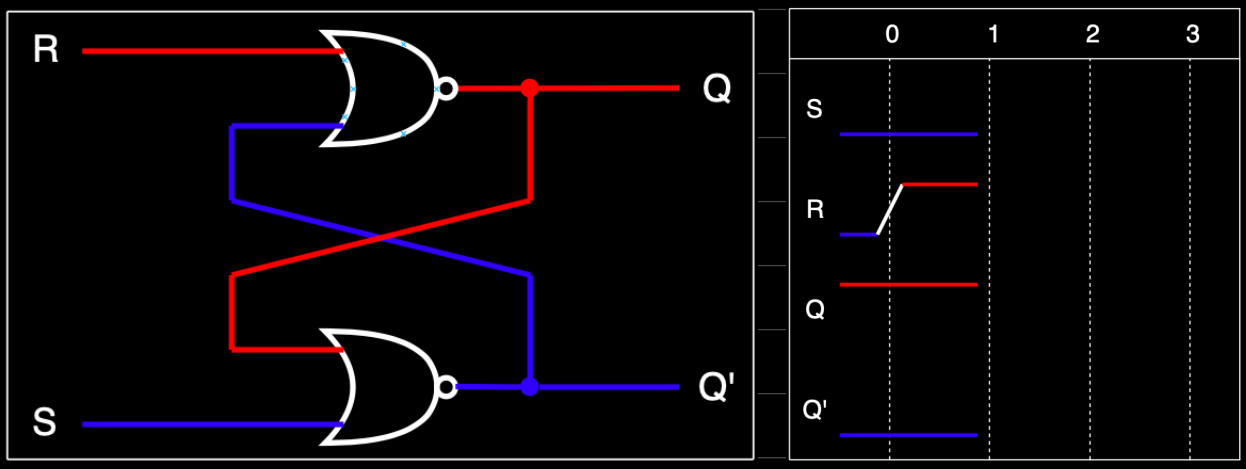
In any case, if used CORRECTLY, i.e. by keeping S/R high for a LONG ENOUGH time when setting/clearing the latch and by NOT keeping S & R high at the same time, then the latch will work AS EXPECTED in ideal as well as real-world circuits.

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

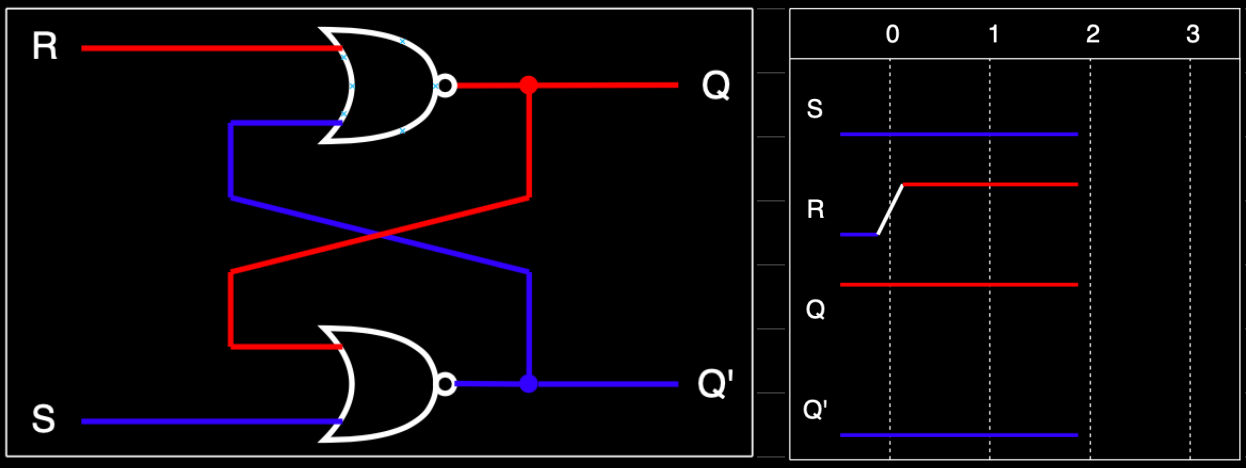


Let R get changed to 1 at t = 0 (in order for the latch to become CLEAR).

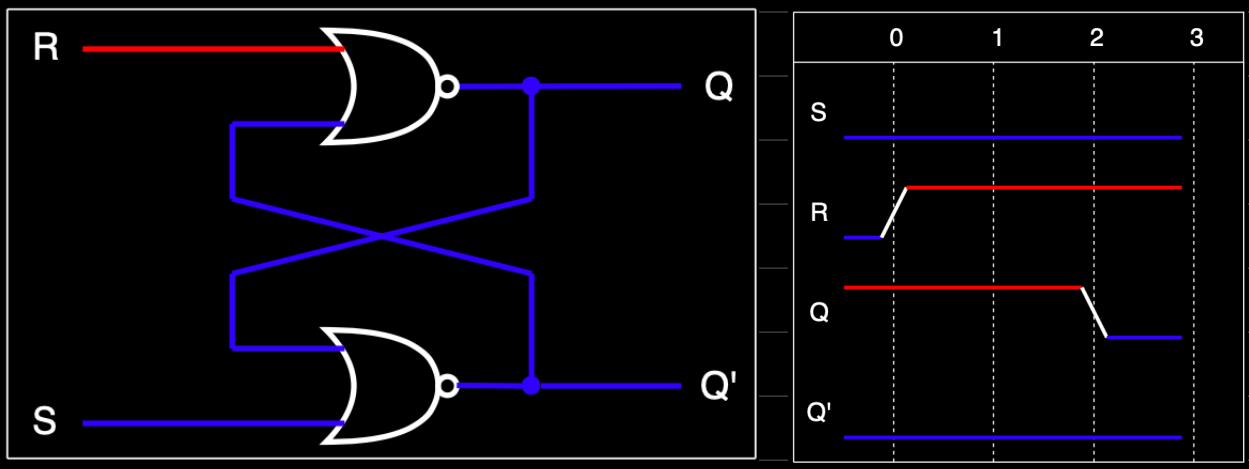
From t = 0 to t = 1 :-



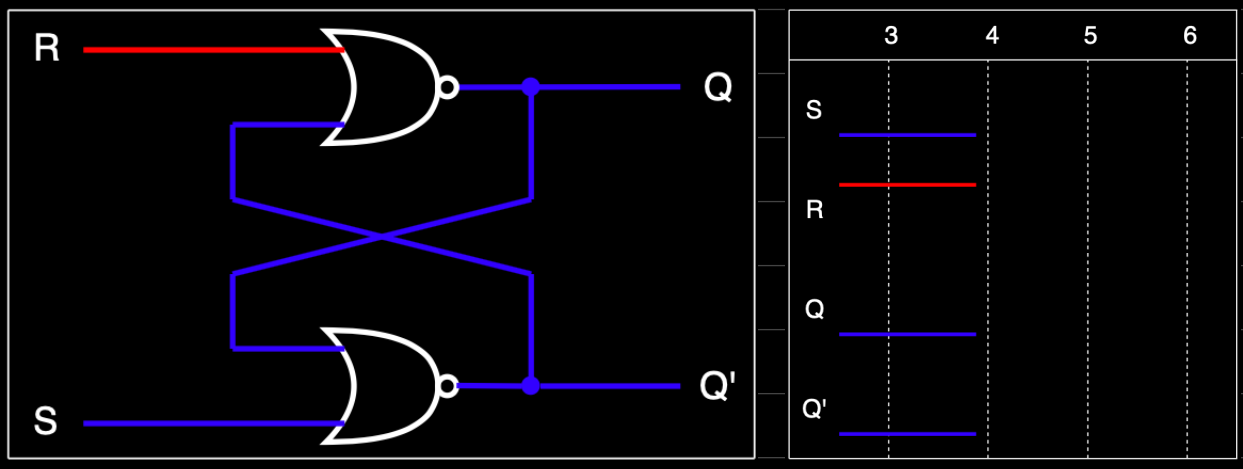
From t = 1 to t = 2 :-



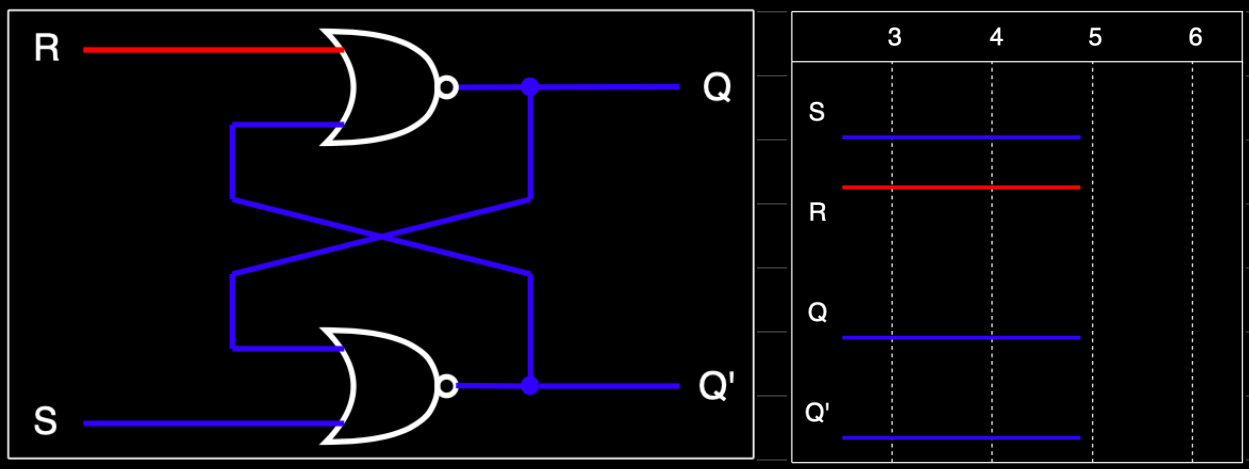
From t = 2 to t = 3 :-



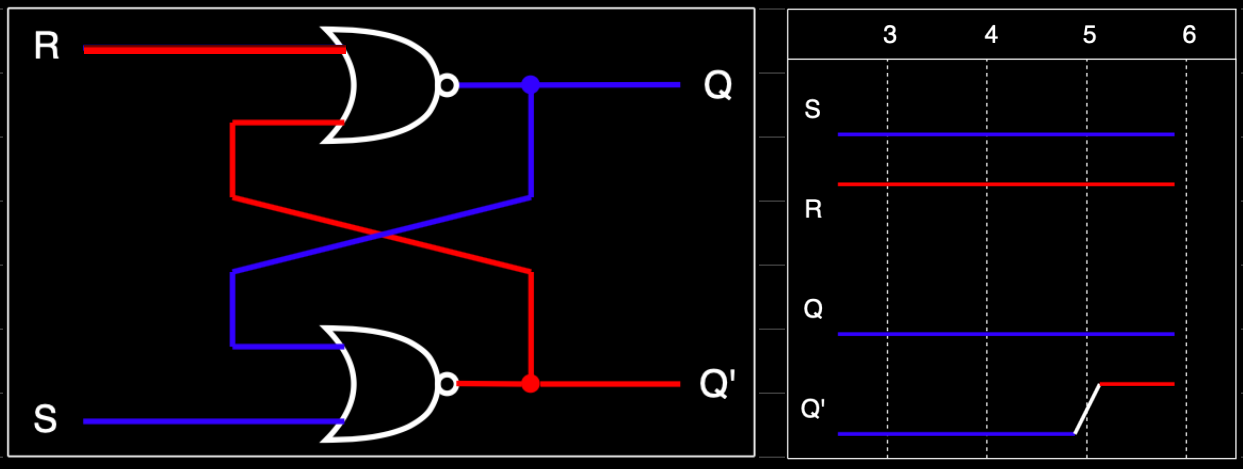
From t = 3 to t = 4 :-



From t = 4 to t = 5 :-



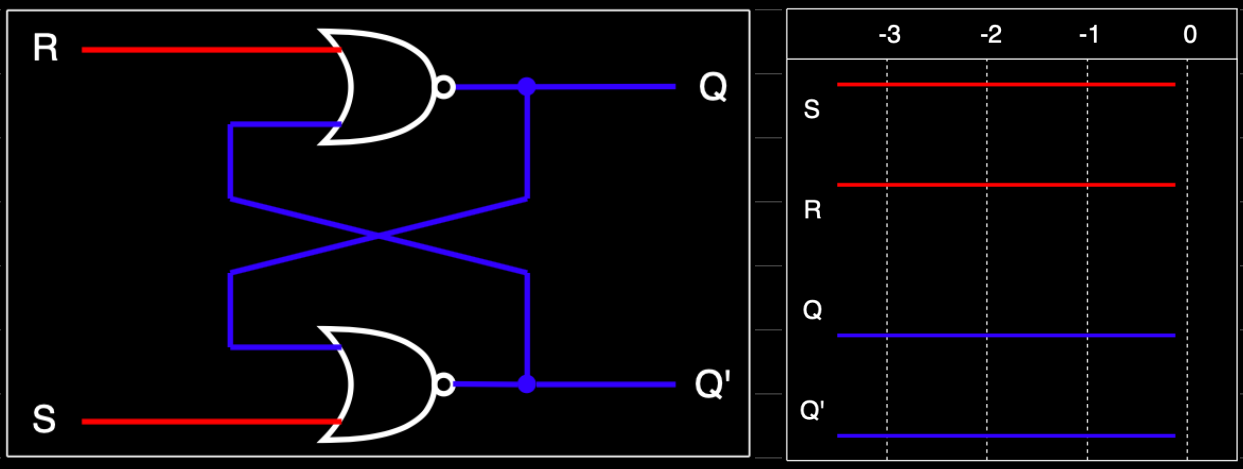
From t = 5 to t = 6 :-



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.

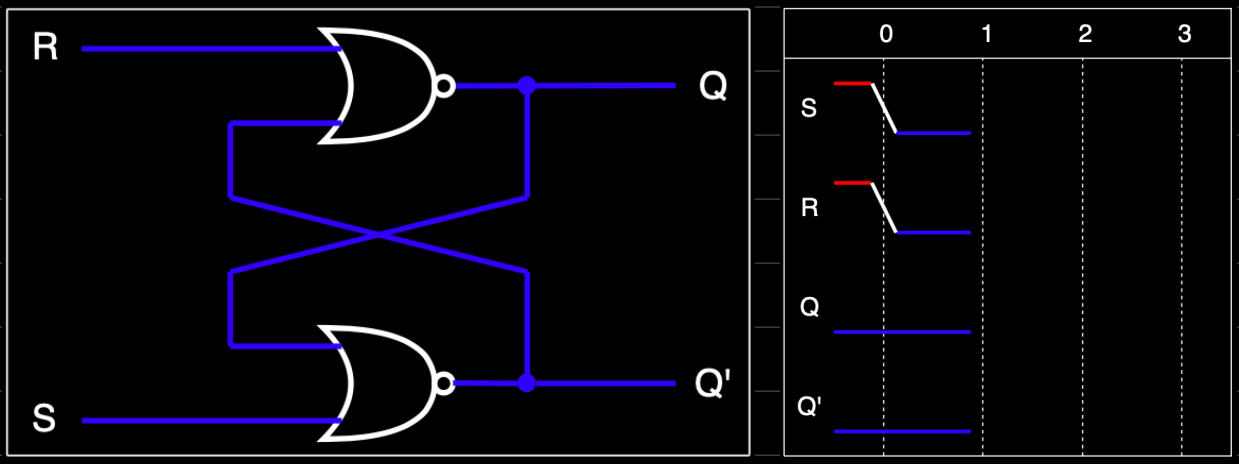
Therefore, if R is NOT maintained at 1 for a LONG ENOUGH time, then similar problems may occur like if S is NOT maintained at 1 for a LONG ENOUGH time when SETTING the latch.

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

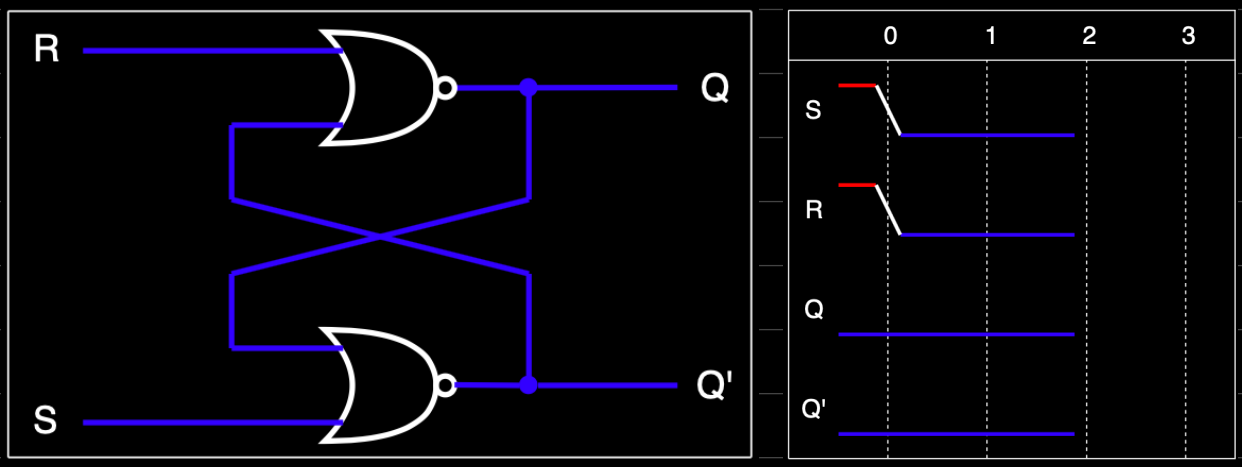


Let S & R both get changed to 0 at t = 0.

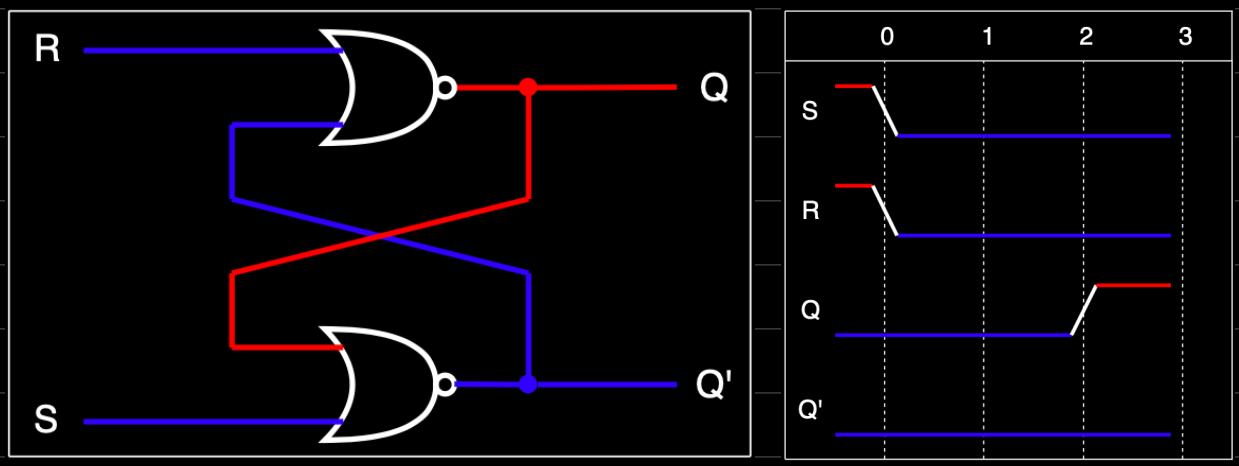
From t = 0 to t = 1 :-



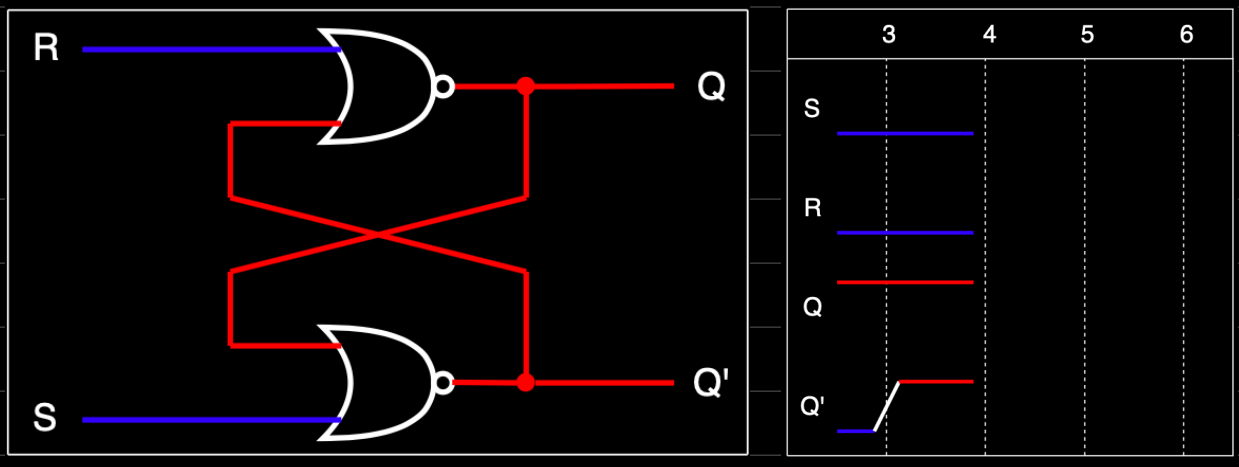
From t = 1 to t = 2 :-



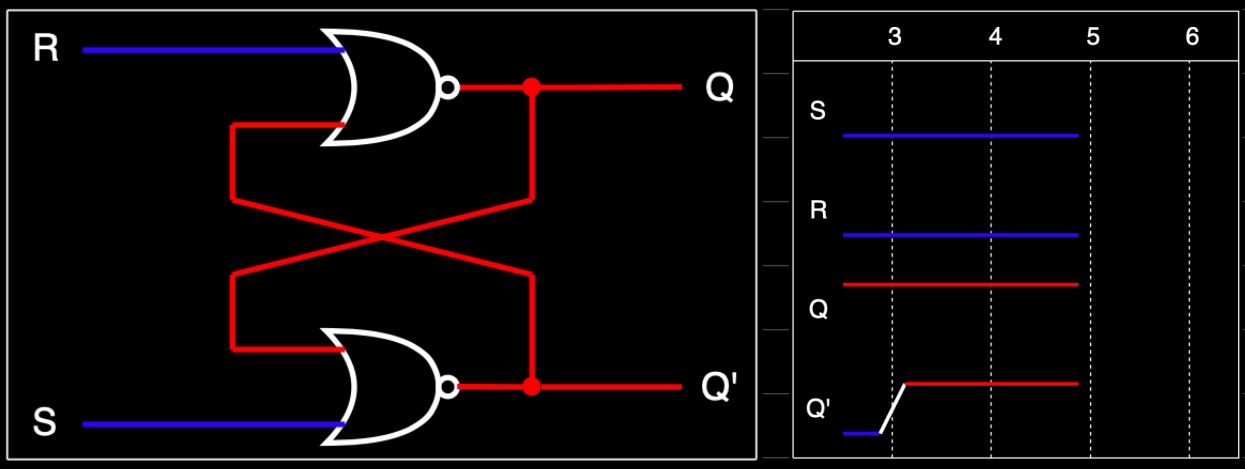
From t = 2 to t = 3 :-



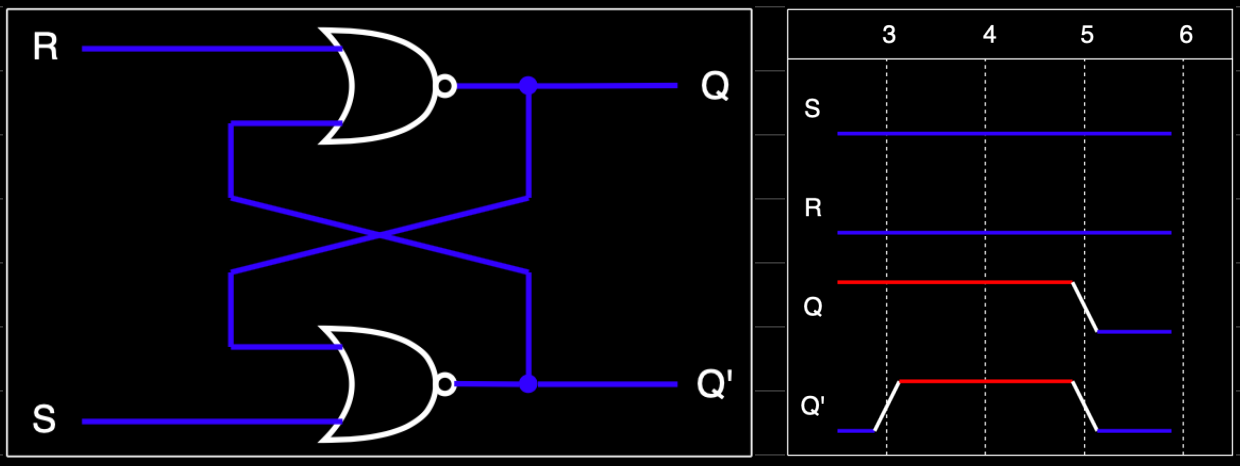
From t = 3 to t = 4 :-



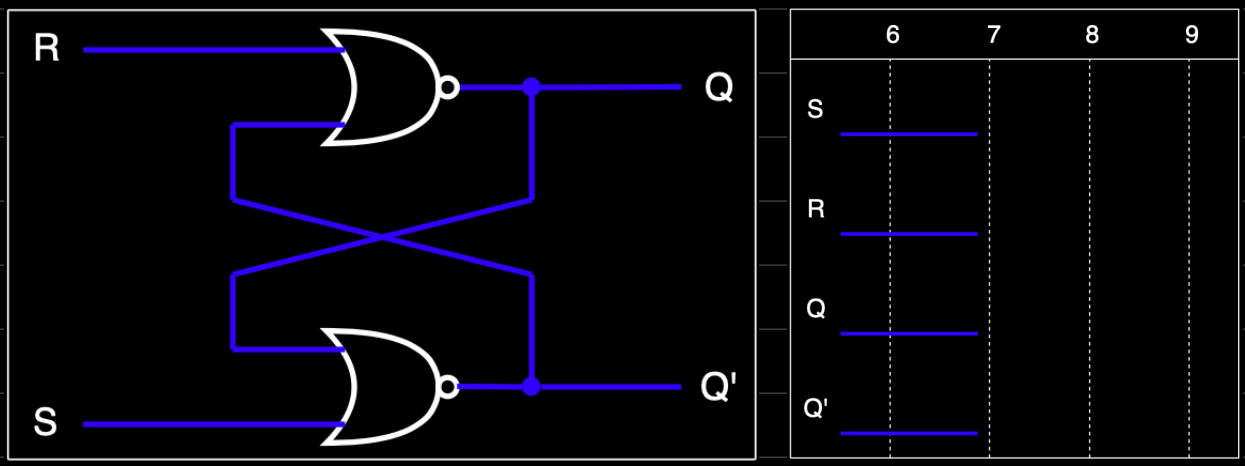
From t = 4 to t = 5 :-



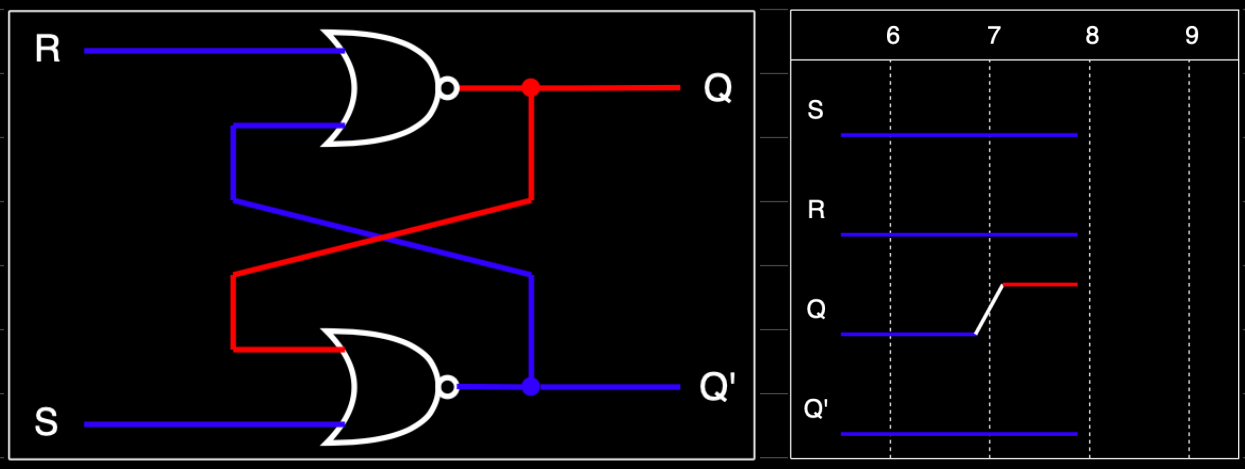
From t = 5 to t = 6 :-



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 (including floating at 1/0 and 0/1), respectively, then no matter the current state of the latch, stable, oscillating between 0 & 1, or even floating (at 0, 1 or somewhere between 0 & 1, or oscillating among 0, 1 and somewhere 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 becomes 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)).

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 (including floating at 1/0 and 0/1), respectively, then no matter the current state of the latch, stable, oscillating between 0 & 1, or even floating (at 0, 1 or somewhere between 0 & 1, or oscillating among 0, 1 and somewhere 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 becomes 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)).

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.

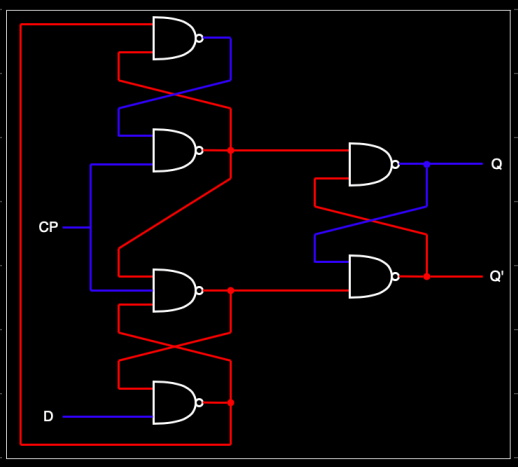
In real-world circuits, the duration of a clock cycle is MUCH MORE than this.

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]

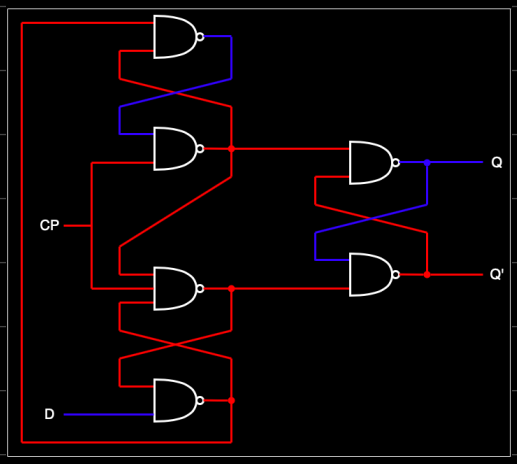
Let the initial STABLE state be CP = 0, D = 0, Q = 0 and Q' = 1 (CLEAR), and let the NEXT bit to be stored be 0 (which is present at the D input at t = 0 and whose SETUP time is completed), then 1, then 1, and then 0.

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).

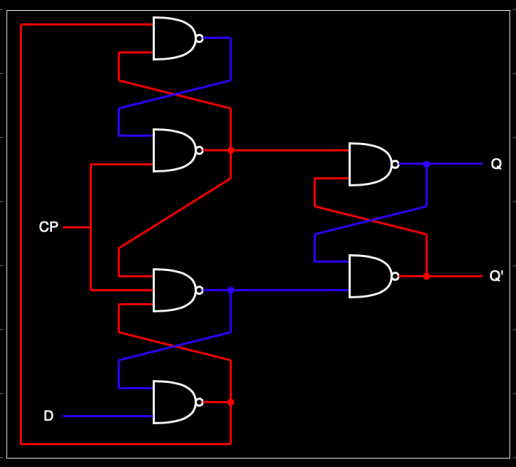
Till t = 0 :-



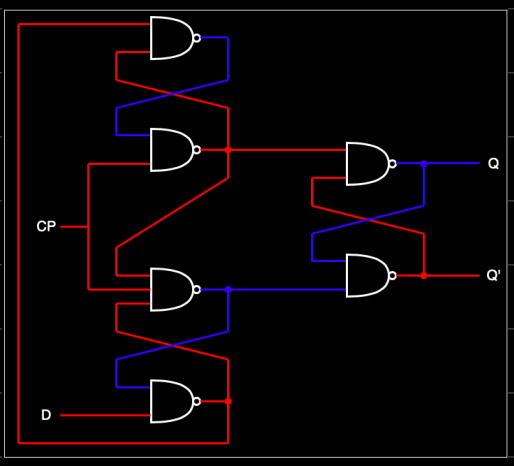
From t = 0 to t = 1 :-



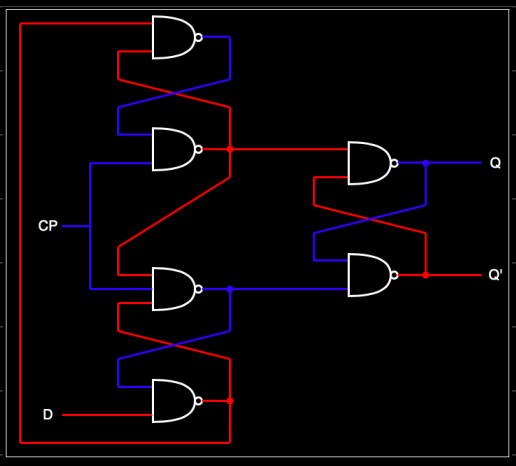
From t = 1 to t = 7 :-



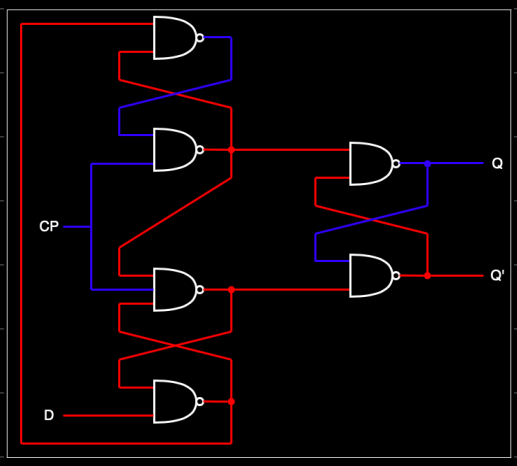
From t = 7 to t = 10 :-



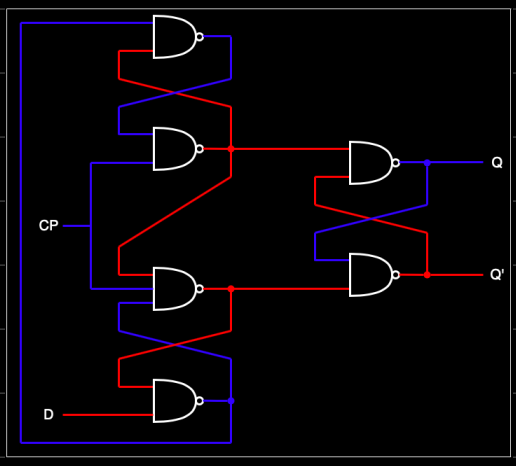
From t = 10 to t = 11 :-



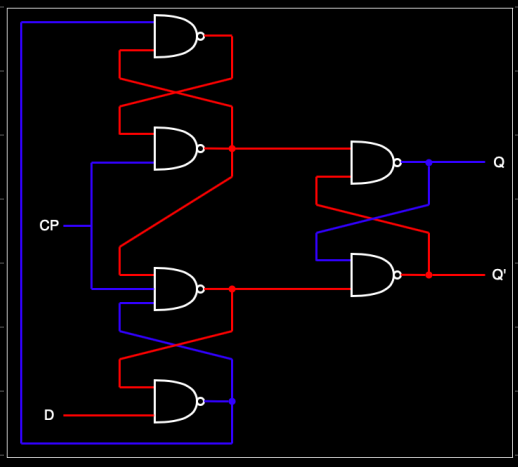
From t = 11 to t = 12 :-



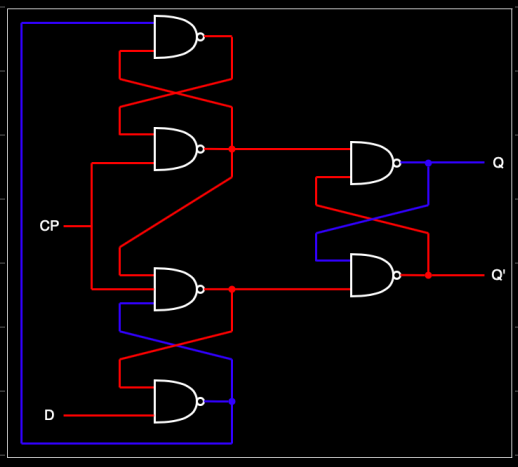
From t = 12 to t = 13 :-



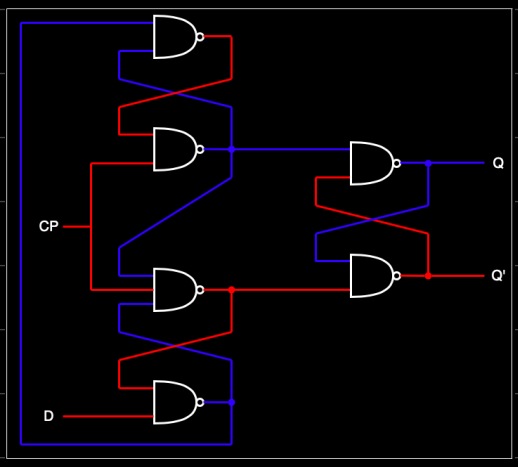
From t = 13 to t = 20 :-



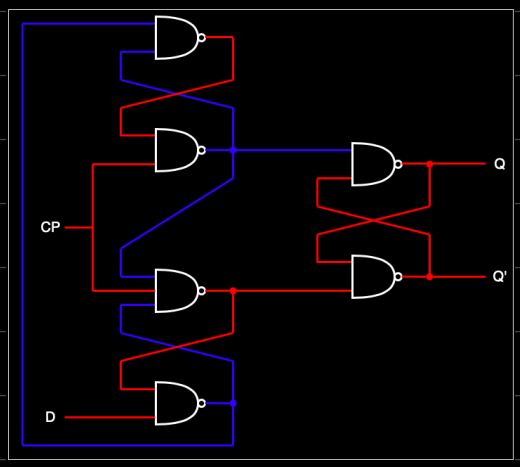
From t = 20 to t = 21 :-



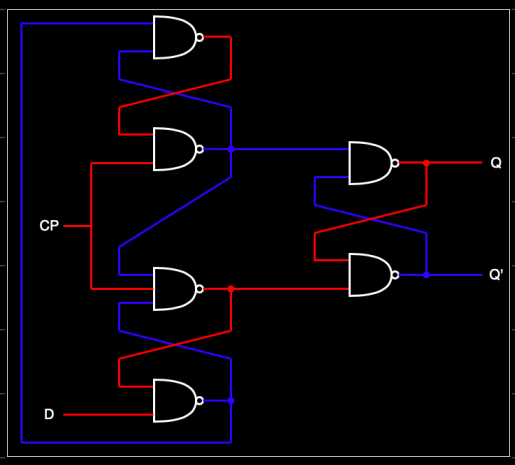
From t = 21 to t = 22 :-



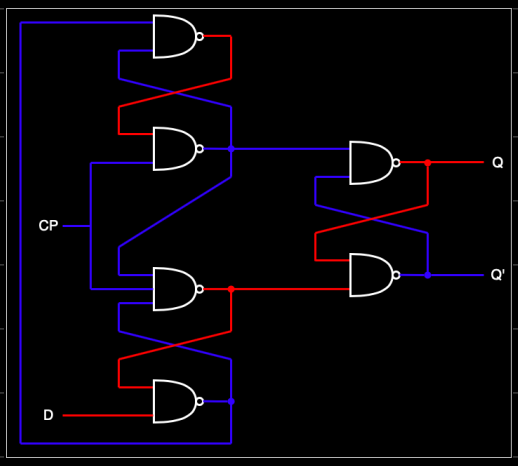
From t = 22 to t = 23 :-



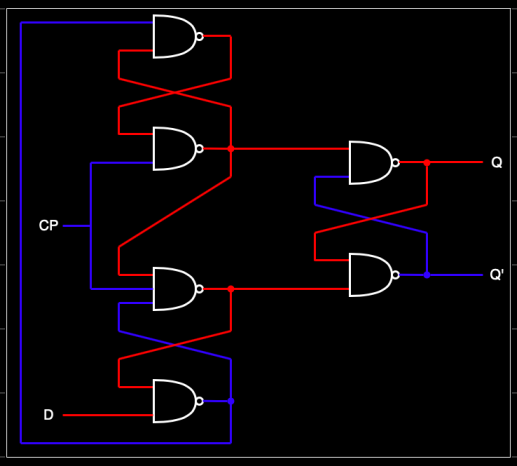
From t = 23 to t = 30 :-



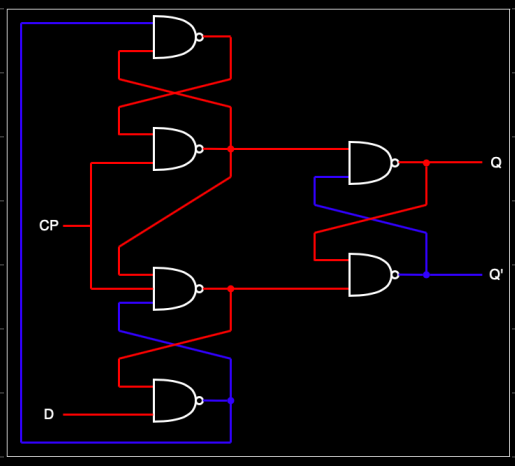
From t = 30 to t = 31 :-



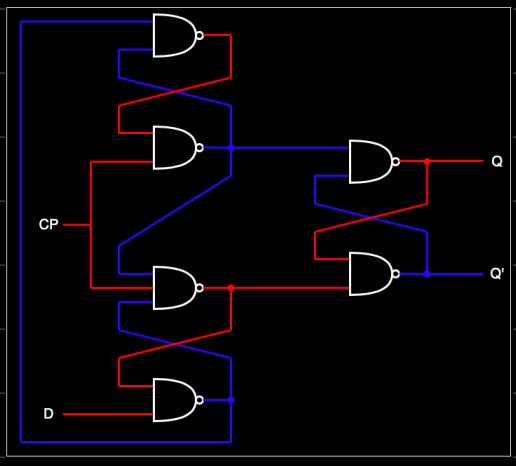
From t = 31 to t = 40 :-



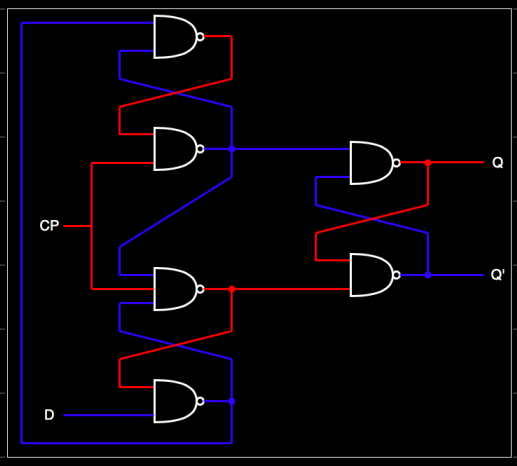
From t = 40 to t = 41 :-



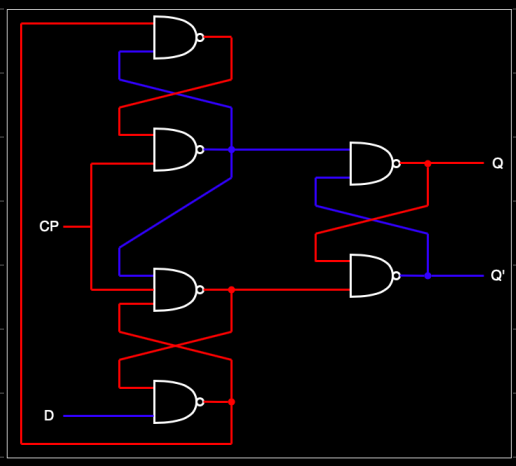
From t = 41 to t = 47 :-



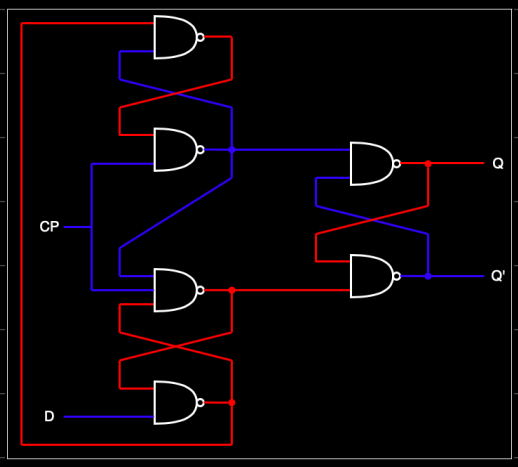
From t = 47 to t = 48 :-



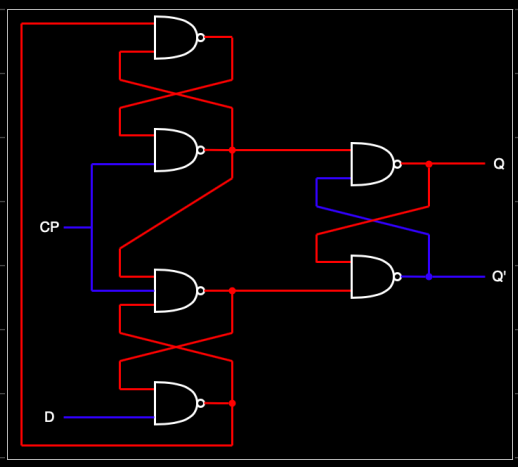
From t = 48 to t = 50 :-



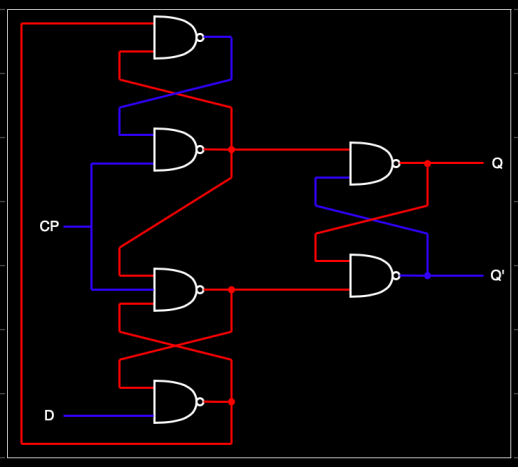
From t = 50 to t = 51 :-



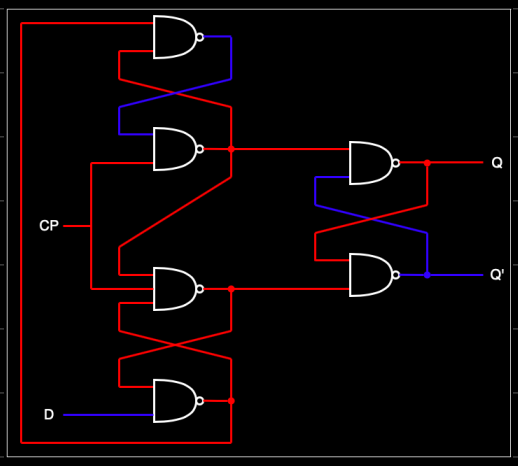
From t = 51 to t = 52 :-



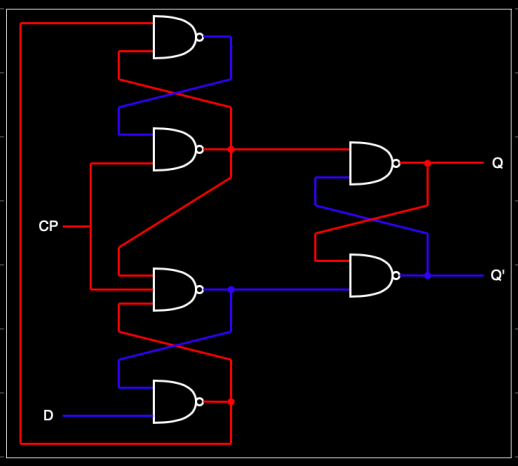
From t = 52 to t = 60 :-



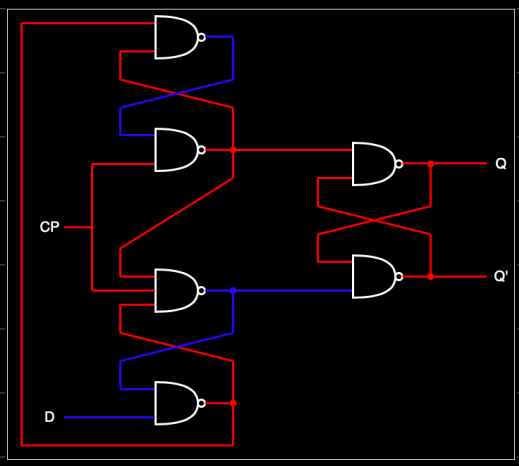
From t = 60 to t = 61 :-



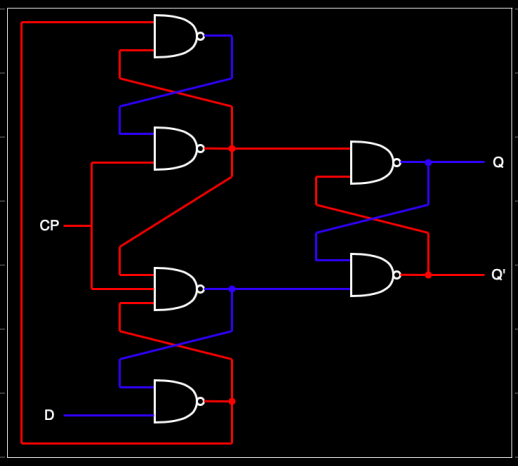
From t = 61 to t = 62 :-



From t = 62 to t = 63 :-



From t = 63 onwards :-

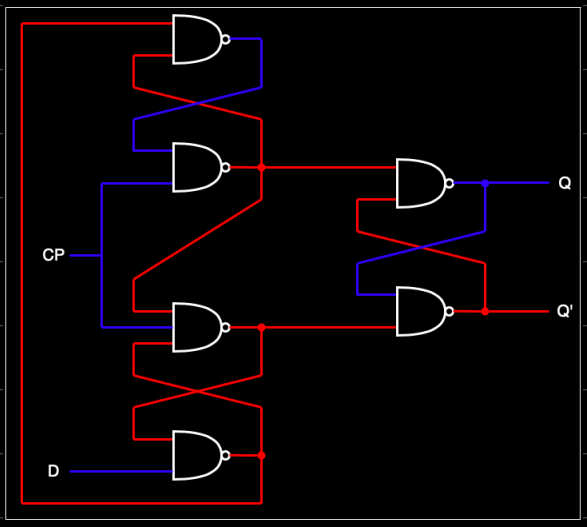


Explanation for the SETUP time and the HOLD time when the flip-flop is getting changed from CLEAR to SET :-

Let the initial STABLE state be

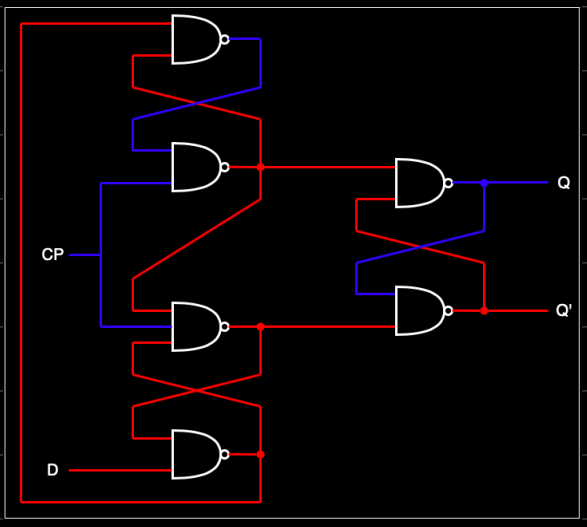
CP = 0, D = 0, Q = 0 and

Q' = 1 (CLEAR).

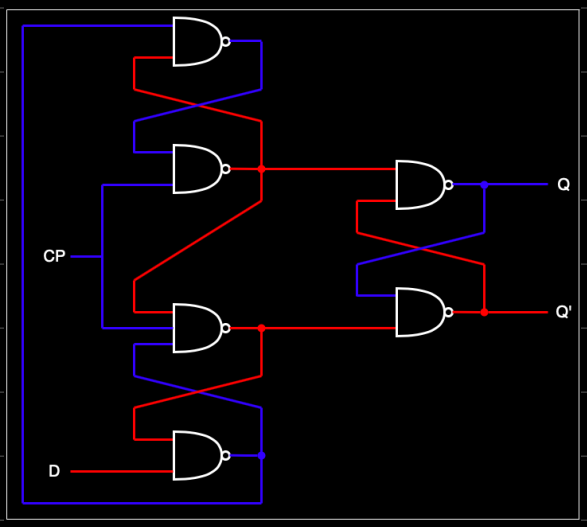


Let D get changed to 1 at t = 0.

From t = 0 to t = 1 –

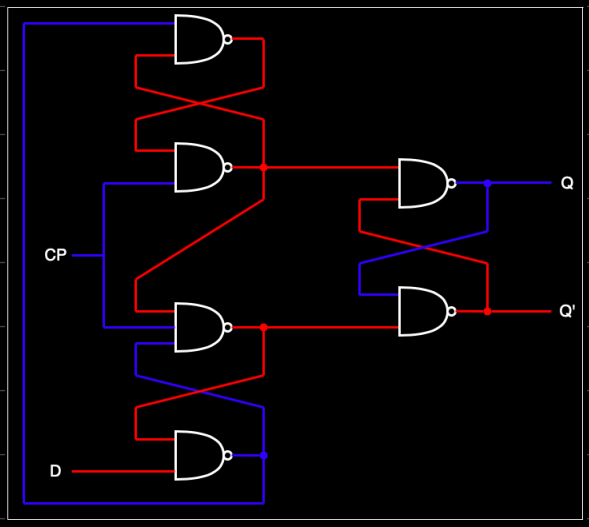


From t = 1 to t = 2 –



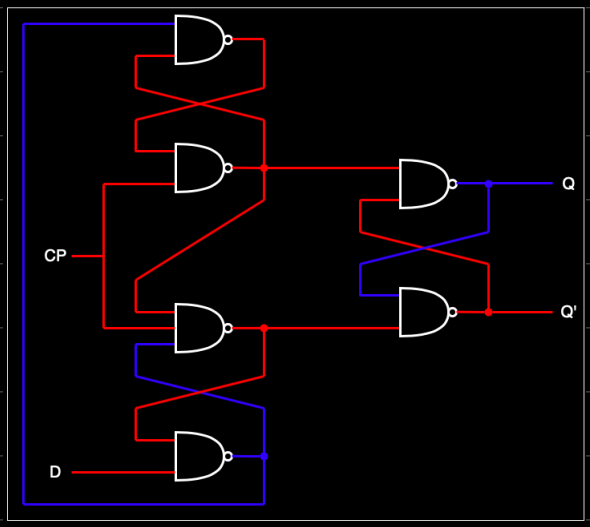
From t = 2 onwards –

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



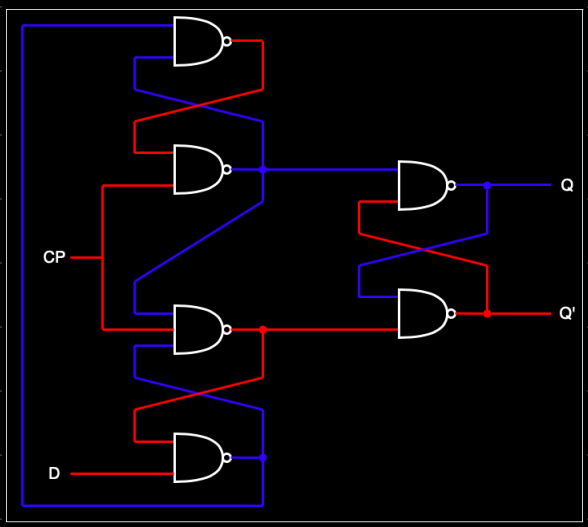
Now, let the initial STABLE state be CP = 0, D = 1, Q = 0 and Q' = 1 (CLEAR) and let CP get changed to 1 at t = 0.

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.



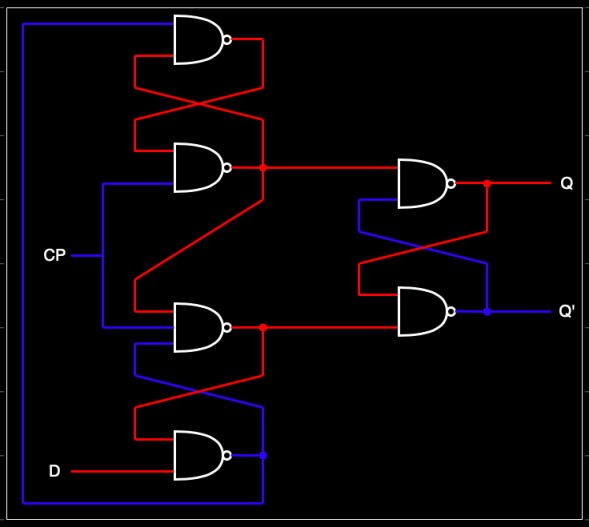
Therefore, the HOLD time is equal to the propagation delay through the UPPER gate to which CP is connected.

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

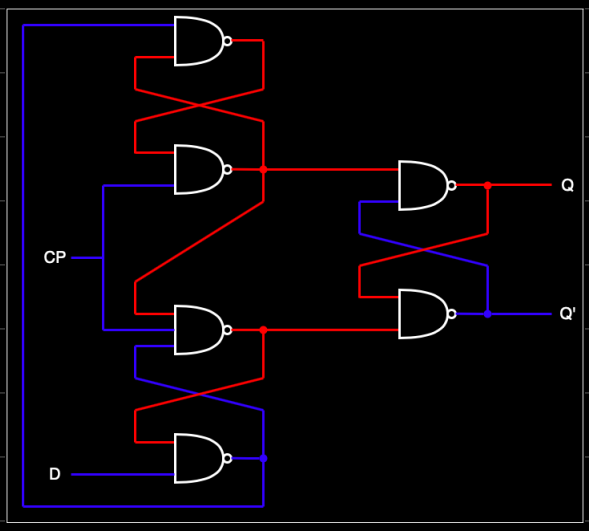
CP = 0, D = 1, Q = 1 and

Q' = 0 (SET).

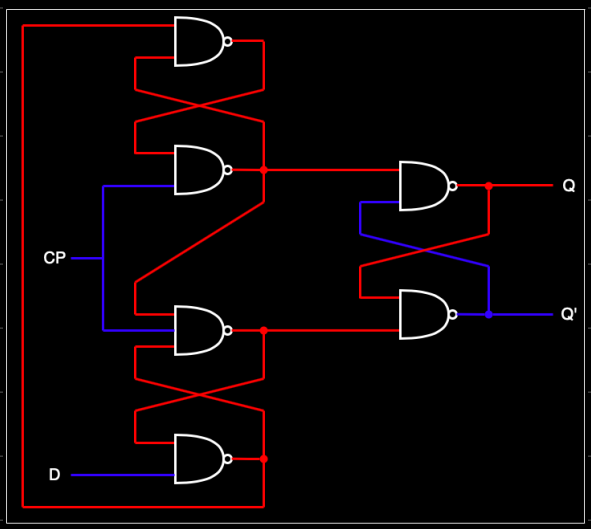


Let D get changed to 0 at t = 0.

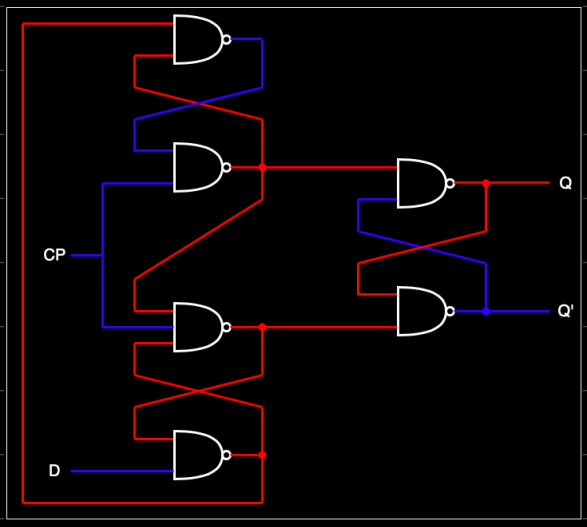
From t = 0 to t = 1 –



From t = 1 to t = 2 –

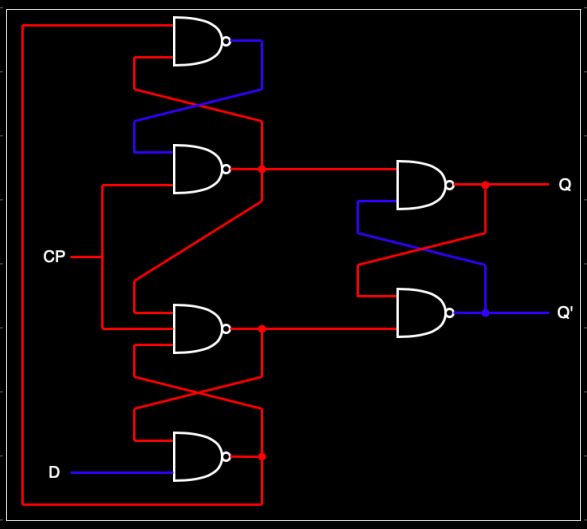


From t = 2 onwards –



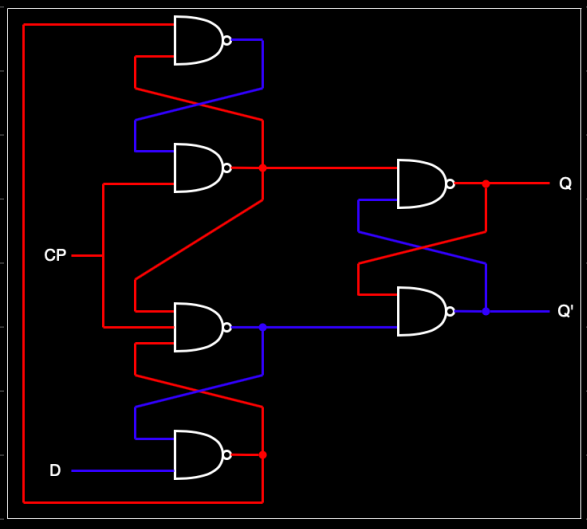
So, the SETUP time is equal to the propagation delays through the BOTTOMMOST gate and then through the TOPMOST gate. In the aforementioned transitions, from t = 50 onwards, it may SEEM like the SETUP time is 1 unit, but it should be noted that the FIRST part of SETUP, i.e. the propagation through the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

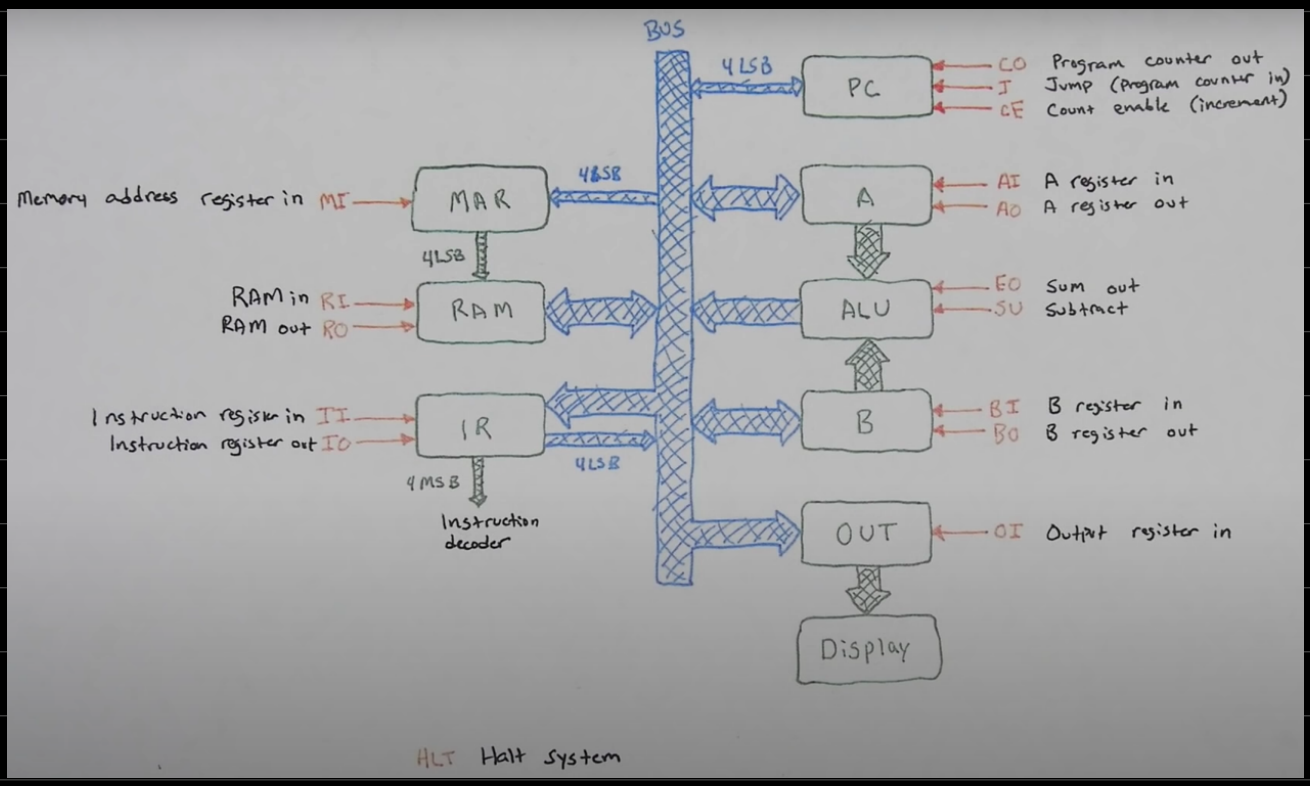
What happens if CP gets changed to 1 before the setup time is completed?

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 16-bit CONTROL WORDS.

From the above diagram, BO (B Register Out) is not used, and is ALWAYS disabled.

Instead of BO, 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, due to those OUTPUTS which are driving these INPUTS FLOATING (at 0, 1 or somewhere between 0 & 1, or oscillating among 0, 1 and somewhere between 0 & 1), 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 remain FLOATING (at 0, 1 or somewhere between 0 & 1, or oscillating among 0, 1 and somewhere between 0 & 1).

For eg., if ONE input of an OR gate becomes 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)).
However, if ONE input of an OR gate becomes 0 (including FLOATING at 0) and the OTHER input becomes FLOATING somewhere between 0 & 1, or if BOTH inputs become FLOATING somewhere between 0 & 1, then after its PROPAGATION delay, its OUTPUT will become FLOATING (at 0, 1 or somewhere between 0 & 1, or oscillating among 0, 1 and somewhere between 0 & 1) if its OUTPUT is NOT being driven by some other component.

If the ENABLE input of a tri-state buffer is 0 (including FLOATING at 0) or FLOATING somewhere between 0 & 1, then the tri-state buffer is said to be DISABLED, whereas if the ENABLE input of a tri-state buffer is 1 (including FLOATING at 1), then the tri-state buffer is said to be ENABLED.
If a tri-state buffer becomes DISABLED, then after its PROPAGATION delay, its OUTPUT will become FLOATING (at 0, 1 or somewhere between 0 & 1, or oscillating among 0, 1 and somewhere between 0 & 1) if its OUTPUT is NOT being driven by some other component.
Moreover, SIMILAR to the aforementioned example of an OR gate, if a tri-state buffer is ENABLED, but its input becomes FLOATING somewhere between 0 & 1, then after its PROPAGATION delay, its OUTPUT will become FLOATING (at 0, 1 or somewhere between 0 & 1, or oscillating among 0, 1 and somewhere between 0 & 1) if its OUTPUT is NOT being driven by some other component.

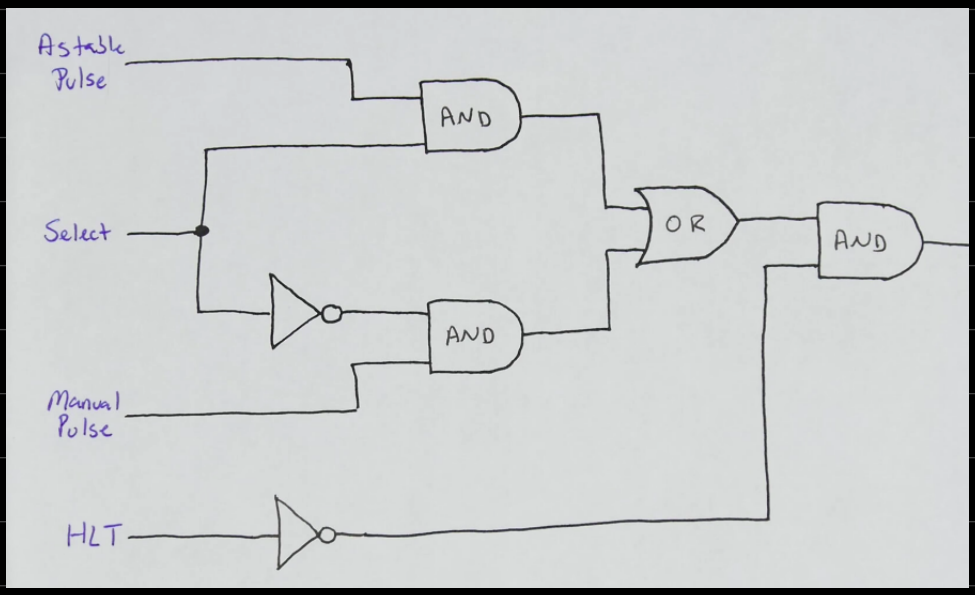
If the input of ONE tri-state buffer which is ENABLED and whose OUTPUT is connected to a BUS wire becomes 0/1 (including FLOATING at 0/1), then after its PROPAGATION delay, the value on that BUS wire will become 0/1, provided that ALL OTHER tri-state buffers whose OUTPUTS are connected to that BUS wire 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 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 of ARBITRARY values getting loaded into GATED LATCHES / FLIP-FLOPS 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 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)).

IMMEDIATELY AFTER the computer is powered up, while the initial PROPAGATIONS have NOT completed, the values on MOST of the wires (except for those which are DIRECTLY fixed to 0's/1's) will be FLOATING (at 0, 1 or somewhere between 0 & 1, or oscillating among 0, 1 and somewhere between 0 & 1), including the CLOCK inputs and the IN signals. However, the values on the BUS wires 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 WILL NOT cause problems in THIS situation as well, as the initial PROPAGATIONS will get completed only a VERY SHORT while AFTER the computer is powered up.

^ Clock

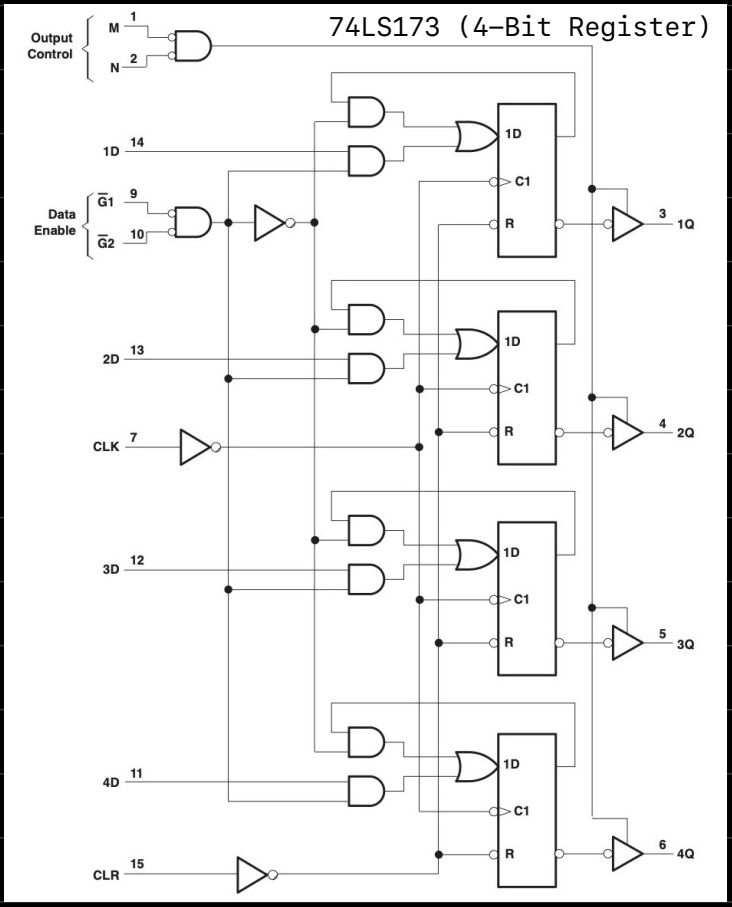


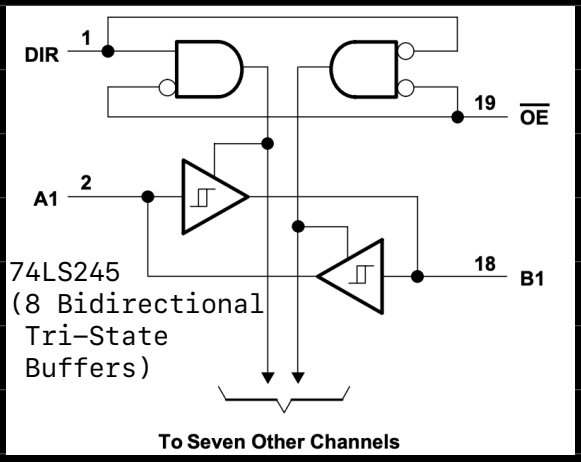
The SELECTION between the ASTABLE (automatic) and the MANUAL pulses is done using a PUSH BUTTON SWITCH which also involves a DEBOUNCING mechanism.

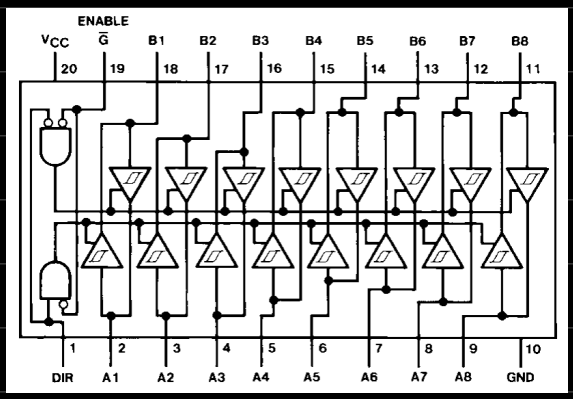
The computer is powered up in the MANUAL pulse mode.
IMMEDIATELY AFTER the computer is powered up, while the initial PROPAGATIONS have NOT completed, the values of the SELECT, the ASTABLE PULSE and the HALT inputs FLOATING (at 0, 1 or somewhere between 0 & 1, or oscillating among 0, 1 and somewhere between 0 & 1), WILL NOT cause problems, as the initial PROPAGATIONS will get completed only a VERY SHORT while AFTER the computer is powered up.

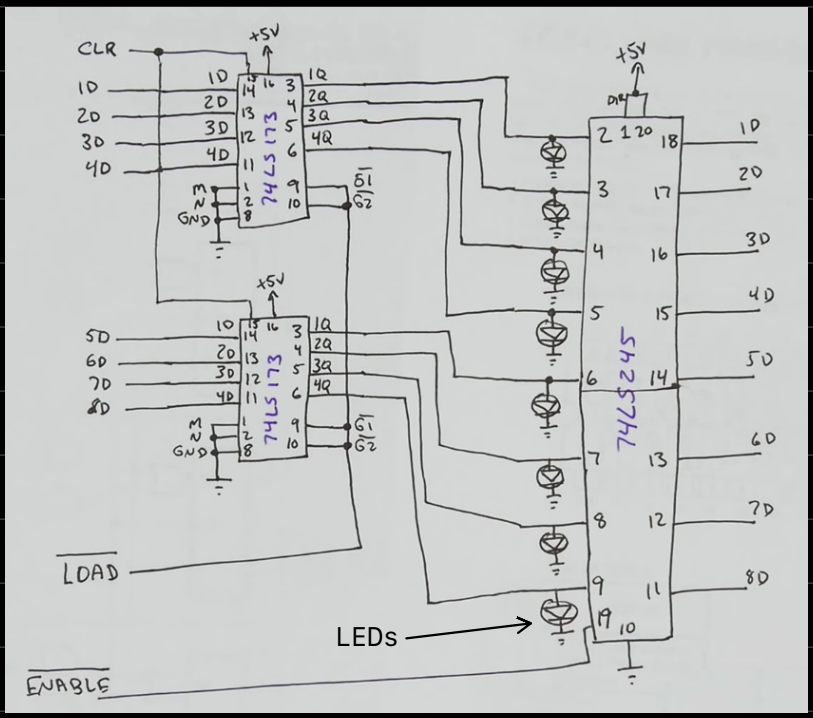
After the program to be run is MANUALLY programmed into the RAM, when the MANUAL clock is switched to the AUTOMATIC clock, then it is ASSUMED that the signals NOT propagating in time will NOT happen, EVEN if the switching is done at a time when the ASTABLE PULSE is nearing its NEGATIVE edge-transition, possibly creating a SHORTER clock pulse than required.

^ A Register (8 Bits), B Register (8 Bits), Memory Address Register (4 Bits), Instruction Register (8 Bits) and Flags Register (2 Bits)









The M and N inputs of EVERY 74LS173 are ALWAYS 0, making the outputs of EVERY 74LS173 ALWAYS enabled, and EXTERNAL tri-state buffers (74LS245) are used.

The DIR input of EVERY 74LS245 is ALWAYS 1/0, fixing the DIRECTION to be from A to B / B to A.

The DIFFERENCE between the MEMORY ADDRESS & FLAGS REGISTERS and the OTHER REGISTERS is that the MEMORY ADDRESS & FLAGS REGISTERS DON'T need to output ANYTHING onto the BUS, and hence they DON'T need EXTERNAL tri-state buffers.

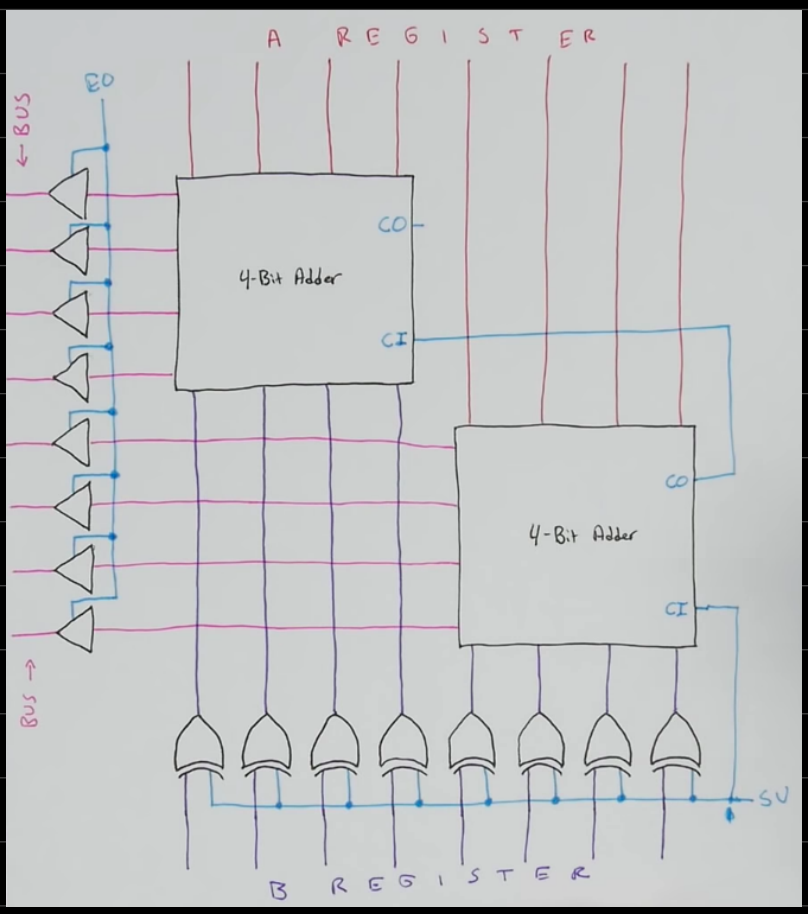
The DIFFERENCE between the INSTRUCTION REGISTER and the A & B REGISTERS is that the 4 MOST SIGNIFICANT output wires from the INSTRUCTION REGISTER's 74LS173 are NOT connected to its 74LS245.
So, these DISCONNECTED inputs of the INSTRUCTION REGISTER's 74LS245 MAY remain FLOATING somewhere between 0 & 1.
In this case, there MAY be a problem if the values on the corresponding wires of the BUS 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 the (Load Immediate) instruction, the A REGISTER would be LOADING the values present on the BUS coming from the INSTRUCTION REGISTER, INCLUDING the 4 MOST SIGNIFICANT bits.

This problem may STILL 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 the DISCONNECTED inputs of the INSTRUCTION REGISTER's 74LS245 MAY remain FLOATING at 1, instead of FLOATING at 0 or somewhere between 0 & 1.
So, DURING the LDI instruction's execution, every such DISCONNECTED input 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 the INSTRUCTION REGISTER's 74LS245 to 0's. 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 general, for ALL ICs, ALL unused INPUTS should be FIXED at 0's or 1's, EVEN FOR unused LOGIC GATES, and ALL unused OUTPUTS should remain DISCONNECTED.

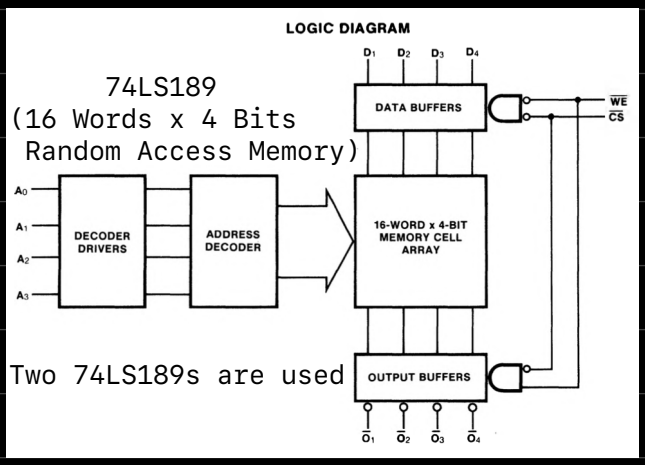
^ Arithmetic Logic Unit



The 4-bit adders correspond to 74LS283's, which use LOOK-AHEAD carry generation.

The tri-state buffers correspond to the ALU's 74LS245.

^ Random Access Memory (16 Words x 8 Bits)

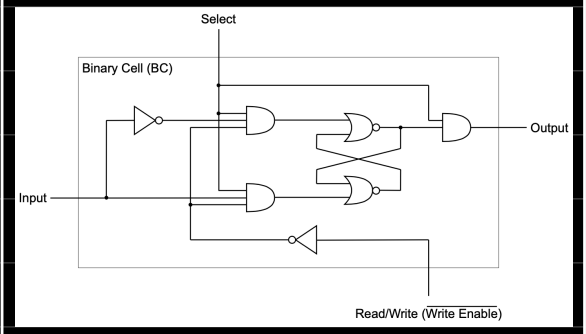


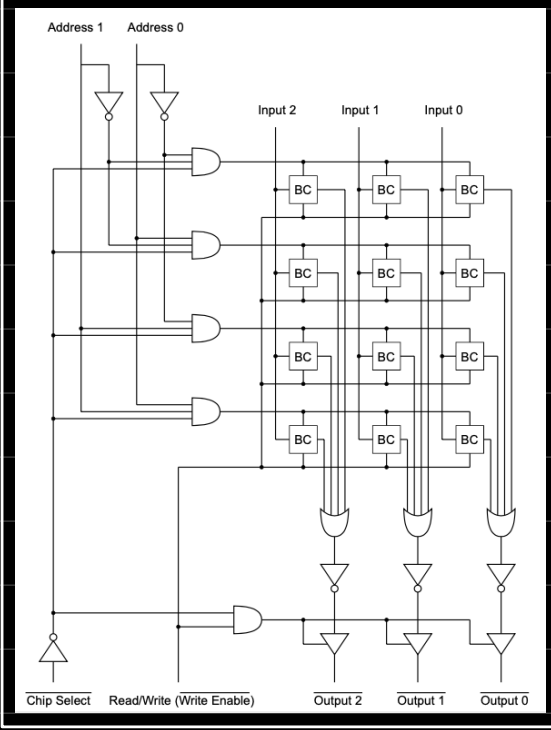
The $\overline{\text{CHIP SELECT}}$ input of EACH 74LS189 is ALWAYS 0, and EXTERNAL tri-state buffers (74LS245) are used.

For some reason, the OUTPUTS of the 74LS189 are INVERTED.

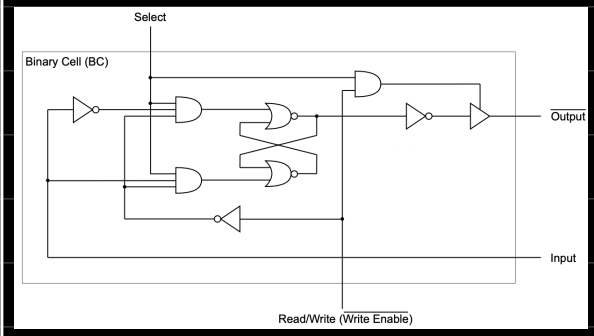
So, EXTERNAL inverters are used to get the correct OUTPUTS.

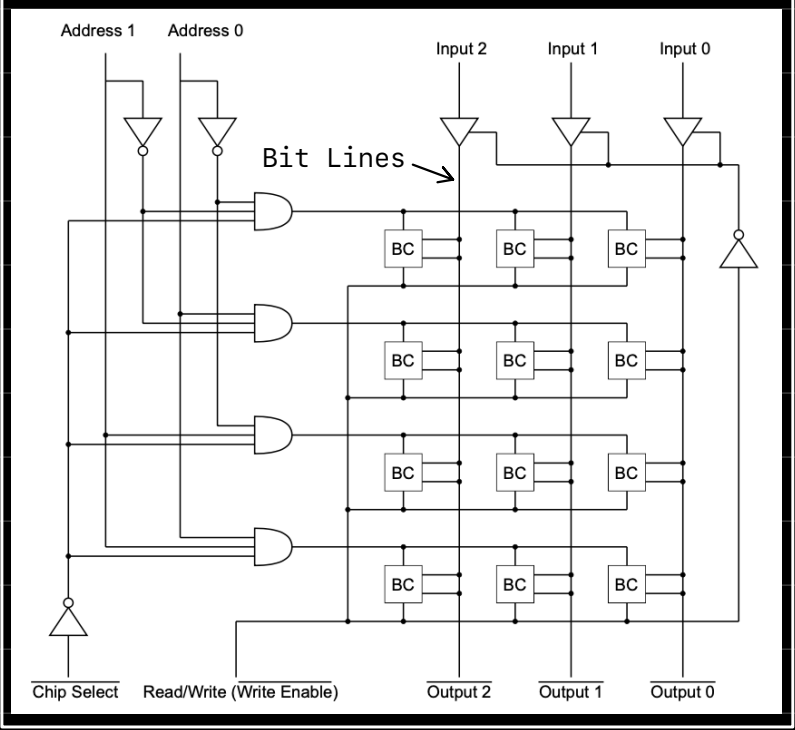
Implementation 1 (4 Words x 3 Bits)

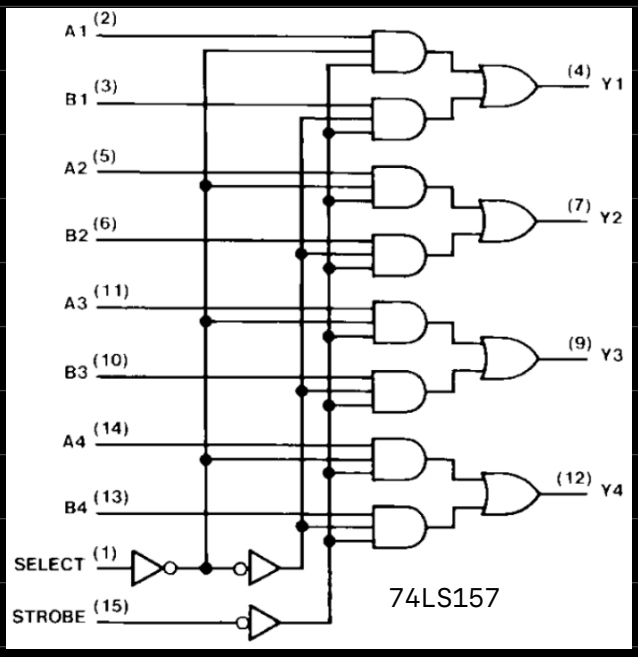




Implementation 2 (4 Words x 3 Bits)







The ADDRESS inputs, the DATA inputs and the $\overline{\text{WRITE ENABLE}}$ input come from the outputs of EXTERNAL MULTIPLEXERS in order to enable switching between MANUAL mode and RUN mode.

The STROBE (ENABLE) input of EVERY 74LS157 is ALWAYS 0, making EVERY 74LS157 ALWAYS enabled.

During MANUAL mode, the ADDRESS and the DATA come from DIP SWITCHES, and the $\overline{\text{WRITE ENABLE}}$ input comes from the PUSH BUTTON. During RUN mode, the ADDRESS comes from the MEMORY ADDRESS REGISTER and the DATA come from the BUS.

The SELECTION between the two modes is done using a PUSH BUTTON SWITCH.

The REGISTERS consist of D FLIP-FLOPS, whereas the RAM consists of GATED D LATCHES (with SLIGHT modifications). So, for the RAM, EDGE-TRIGGERING is achieved using the RESISTOR-CAPACITOR method.

Also, since the RAM DOESN'T have any LOAD input, therefore the CLOCK input (after using the RESISTOR-CAPACITOR method) is NANDed with the RAM IN signal to become the $\overline{\text{WRITE ENABLE}}$ input of the RAM during RUN mode.

Due to DIFFERENT propagation delays of LOGIC GATES,

1. For implementation 1, when the ADDRESS changes while $\overline{\text{WRITE ENABLE}}$ is 1, BINARY CELLS from DIFFERENT words MAY get selected SIMULTANEOUSLY for a VERY SHORT amount of time.

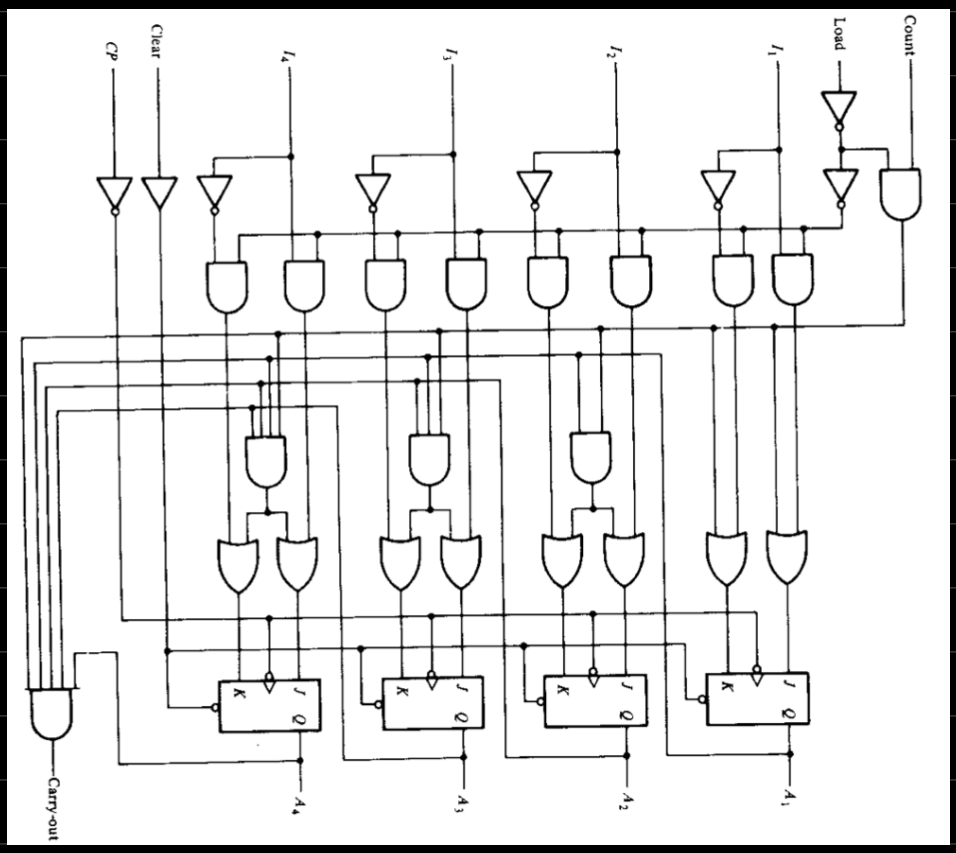
   For implementation 2, such a situation MAY cause CONTENTION on the BIT LINES for a VERY SHORT amount of time, NOT causing any damage to the BIT LINES.

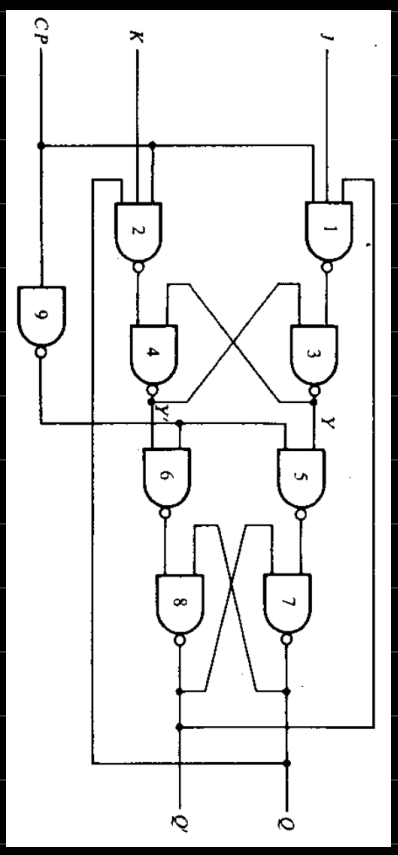
2. For implementation 1, when the ADDRESS changes while $\overline{\text{WRITE ENABLE}}$ is 1, it MAY happen that NO BINARY CELL is selected for a VERY SHORT amount of time.
   For implementation 2, such a situation MAY cause the values on the BIT LINES to be FLOATING (at 0, 1 or somewhere between 0 & 1, or oscillating among 0, 1 and somewhere between 0 & 1) for a VERY SHORT amount of time.
3. For implementation 2, when $\overline{\text{WRITE ENABLE}}$ is switched from 1 to 0, the tri-state buffers at the INPUTS MAY become ENABLED SLIGHTLY before the tri-state buffers in the BINARY CELLS become DISABLED, possibly causing CONTENTION on the BIT LINES. Similarly, when $\overline{\text{WRITE ENABLE}}$ is switched from 0 to 1, the tri-state buffers at the INPUTS MAY become DISABLED SLIGHTLY after the tri-state buffers in the BINARY CELLS become ENABLED, possibly causing CONTENTION on the BIT LINES.
   However, such situations would last ONLY for VERY SHORT amounts of time, NOT causing any damage to the BIT LINES.
4. For implementation 2, when $\overline{\text{WRITE ENABLE}}$ is switched from 1 to 0, the tri-state buffers at the INPUTS MAY become ENABLED SLIGHTLY after the tri-state buffers in the BINARY CELLS become DISABLED, possibly causing the values on the BIT LINES to be FLOATING (at 0, 1 or somewhere between 0 & 1, or oscillating among 0, 1 and somewhere between 0 & 1) for a VERY SHORT amount of time. However, ONCE the tri-state buffers at the INPUTS become ENABLED, CORRECT values will get written into the latches, even though the INPUTS of the latches were possibly FLOATING previously. This is 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. Similarly, when $\overline{\text{WRITE ENABLE}}$ is switched from 0 to 1, the tri-state buffers at the INPUTS MAY become DISABLED SLIGHTLY before the tri-state buffers in the BINARY CELLS become ENABLED, possibly causing FLOATING values to get written to the latches. However, this problem CAN be solved by ENSURING that such a situation does NOT occur, for eg., by adding BUFFERS or an EVEN number of INVERTERS to the RIGHT-MOST wire in order to INCREASE the propagation delay.

^ Program Counter (4 Bits) and Step Counter (4 Bits)

The PROGRAM COUNTER and the STEP COUNTER are POSITIVE edge-triggered.

Such a counter can be constructed using NEGATIVE edge-triggered MASTER-SLAVE flip-flops by INTERNALLY inverting the clock input.





The left diagram shows a POSITIVE edge-triggered 4-bit SYNCHRONOUS counter (similar to 74LS161), and the right diagram shows the NEGATIVE edge-triggered MASTER-SLAVE flip-flops used in the counter.

As opposed to for LOGIC GATES, where a SMALL CIRCLE means that the corresponding input/output is INVERTED, a SMALL CIRCLE in the CLOCK input of a FLIP-FLOP DOESN'T mean that the CLOCK input is INVERTED. It ONLY specifies the type of the FLIP-FLOP (POSITIVE or NEGATIVE edge-triggered).

For the PROGRAM COUNTER, EXTERNAL tri-state buffers (74LS245) are used.
Similar to the INSTRUCTION REGISTER, the 4 MOST SIGNIFICANT output wires from the PROGRAM COUNTER's 74LS161 are also NOT connected to its 74LS245.
Instead, the inputs of the 4 MOST SIGNIFICANT tri-state buffers of the PROGRAM COUNTER's 74LS245 are FIXED to 0's, just like they SHOULD'VE been for the INSTRUCTION REGISTER as well.

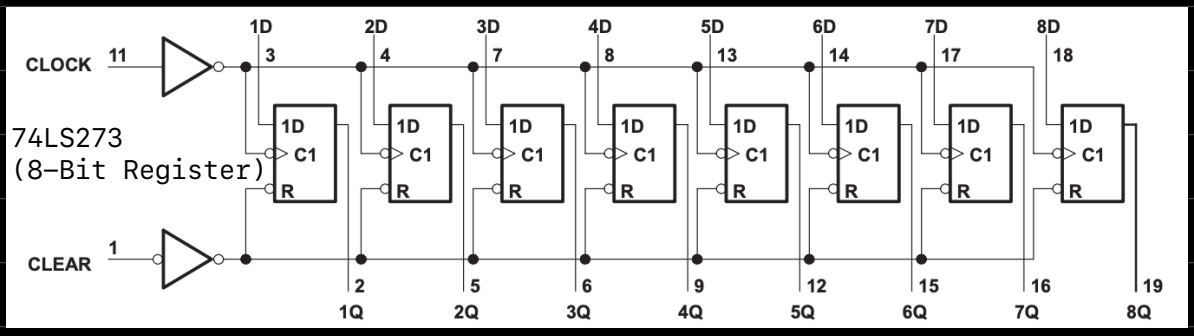
The STEP COUNTER DOESN'T need to output ANYTHING onto the BUS, and hence it DOESN'T need EXTERNAL tri-state buffers.

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

The COUNT (COUNT ENABLE) and LOAD (JUMP) inputs of the PROGRAM COUNTER are controlled via the CONTROL UNIT, whereas for the STEP COUNTER, the COUNT input is FIXED to 1 and the LOAD input is FIXED to 0.

A DECODER having INVERTED outputs (74LS138) is used along with the STEP COUNTER (to RESET it when required) because its CLEAR input is ACTIVE LOW.

^ Output Register (8 Bits)



The OUTPUT REGISTER DOESN'T need to output ANYTHING onto the BUS, and hence it DOESN'T need EXTERNAL tri-state buffers.

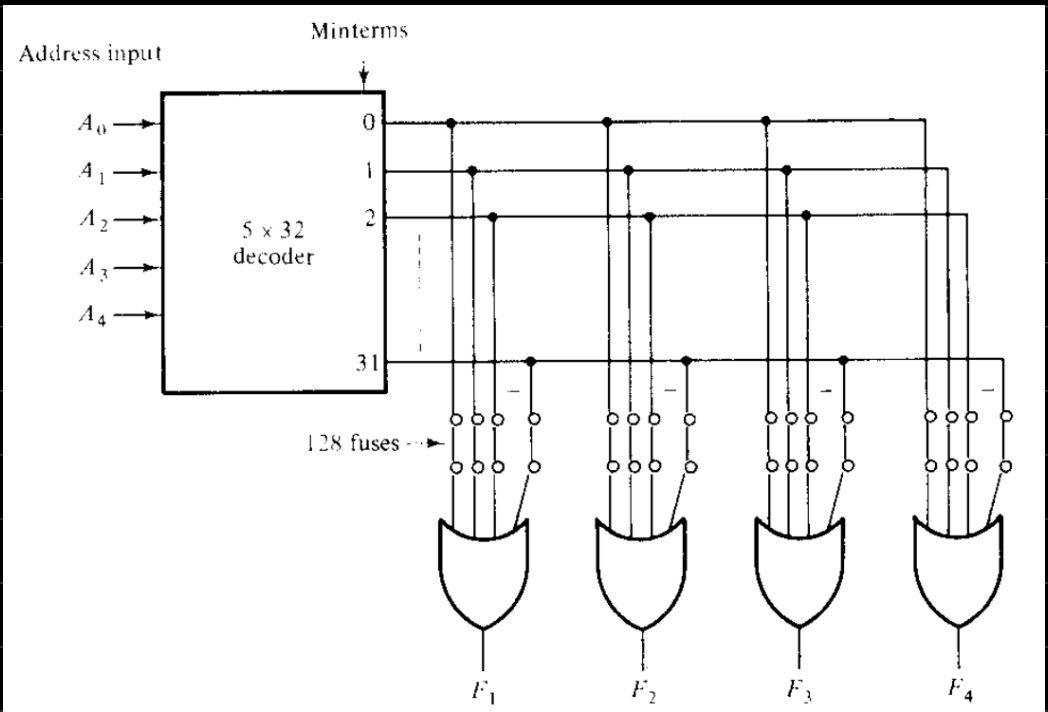
Also, as OPPOSED to the 74LS173, the 74LS273 DOESN'T have any LOAD input, therefore the MASTER CLOCK is ANDed with the OUTPUT REGISTER IN signal to become the CLOCK input of the OUTPUT REGISTER.

The LOGIC DIAGRAM of the 74LS173 is MORE complicated than that of the 74LS273, as it involves EXTRA logic to provide a LOAD input.

Instead of using the 74LS173 for the A REGISTER, the B REGISTER, the MEMORY ADDRESS REGISTER, the INSTRUCTION REGISTER and the FLAGS REGISTER, the 74LS273 could've ALSO been used instead by ANDing the MASTER CLOCK with the corresponding REGISTER IN signal, as the OUTPUTS of those REGISTERS are ALWAYS enabled anyway.
This is because EVEN THOUGH the insertion of an AND gate would produce PROPAGATION DELAYS between the MASTER CLOCK and the CLOCK inputs of the FLIP-FLOPS, such PROPAGATION DELAYS would cause NO problems.

^ Control Unit

The INSTRUCTION REGISTER's, the STEP COUNTER's and the FLAGS REGISTER's outputs determine the ADDRESS of each EEPROM of the CONTROL UNIT.
After each EEPROM has been PROGRAMMED, its $\overline{\text{WRITE ENABLE}}$ input is FIXED to 1, and its $\overline{\text{CHIP ENABLE}}$ & $\overline{\text{OUTPUT ENABLE}}$ inputs are FIXED to 0's, making each EEPROM ALWAYS enabled.



The above diagram shows a 32-Word x 4-Bit READ-ONLY MEMORY (ROM).
It is ASSUMED that for the OR gates, an INPUT whose link is BROKEN, i.e. a DISCONNECTED input, gets converted to 0, for both IDEAL and REAL-WORLD circuits.
The internal construction of this ROM is SIMILAR to the aforementioned Implementation 1 of the RAM.

An EEPROM's internal construction MAY be similar to the aforementioned Implementation 1 or Implementation 2 (using tri-state buffers) of the RAM, or it MAY be something else ENTIRELY.

For those EEPROMs whose internal constructions are SIMILAR to the aforementioned implementations of the RAM, due to DIFFERENT propagation delays of LOGIC GATES,

1. For implementation 1, when the ADDRESS changes, MULTIPLE decoder outputs MAY get enabled SIMULTANEOUSLY for a VERY SHORT amount of time.
   For implementation 2, such a situation MAY cause CONTENTION on the OUTPUT LINES for a VERY SHORT amount of time, NOT causing any damage to the OUTPUT LINES.
2. For implementation 1, when the ADDRESS changes, it MAY happen that NO decoder output is selected for a VERY SHORT amount of time.
   For implementation 2, such a situation MAY cause the values on the OUTPUT LINES to be FLOATING (at 0, 1 or somewhere between 0 & 1, or oscillating among 0, 1 and somewhere between 0 & 1) for a VERY SHORT amount of time.

However, there would be NO problems of ARBITRARY values getting loaded into GATED LATCHES / FLIP-FLOPS due to THESE situations, as the CLOCK input 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 and the CONTROL signals (even FLOATING (at 0, 1 or somewhere between 0 & 1, or oscillating among 0, 1 and somewhere between 0 & 1)).

For eg., 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 becomes DISABLED according to the NEXT CONTROL WORD, then the propagation 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 becomes ENABLED according to that NEXT CONTROL WORD, POSSIBLY causing the two components to output their contents onto the BUS SIMULTANEOUSLY, causing BUS CONTENTION.
However, such a situation would last ONLY for a VERY SHORT amount of time, NOT causing any damage to the BUS.

IMMEDIATELY AFTER the computer is powered up, while the initial PROPAGATIONS have NOT completed, the values on MOST of the wires (except for those which are DIRECTLY fixed to 0's/1's) will be FLOATING (at 0, 1 or somewhere between 0 & 1, or oscillating among 0, 1 and somewhere between 0 & 1), including the CONTROL signals (i.e. the OUTPUTS of the CONTROL UNIT's EEPROMs), which MAY cause arbitrary INVALID CONTROL WORDS to get generated one after another.
However, such a situation WILL NOT cause problems, as the initial PROPAGATIONS will get completed only a VERY SHORT while AFTER the computer is powered up.

Now, EVEN AFTER the initial PROPAGATIONS get completed, the OUTPUT of every GATED LATCH / FLIP-FLOP will be ARBITRARY (0, 1, OSCILLATING between 0 & 1 or even FLOATING somewhere between 0 & 1).

So, if some OUTPUTS of the INSTRUCTION REGISTER, the STEP COUNTER or the FLAGS REGISTER are OSCILLATING between 0 & 1, then arbitrary, although VALID, CONTROL WORDS MAY get generated one after another EVEN AFTER the initial PROPAGATIONS get completed.

In the WORST case, if all OUTPUTS of the INSTRUCTION REGISTER, the STEP COUNTER and the FLAGS REGISTER are FLOATING somewhere between 0 & 1, then all CONTROL signals (i.e. all OUTPUTS of the CONTROL UNIT's EEPROMs) will become FLOATING (at 0, 1 or somewhere between 0 & 1, or oscillating among 0, 1 and somewhere between 0 & 1), which MAY cause arbitrary INVALID CONTROL WORDS to get generated one after another.
Due to this, MULTIPLE OUT signals may become ENABLED SIMULTANEOUSLY for a LONG TIME, causing BUS CONTENTION and DAMAGING the BUS WIRES.

This is because, for eg., if BOTH inputs of an OR gate become FLOATING somewhere between 0 & 1, then after its PROPAGATION delay, its OUTPUT will become FLOATING (at 0, 1 or somewhere between 0 & 1, or oscillating among 0, 1 and somewhere between 0 & 1) if its OUTPUT is NOT being driven by some other component.

Similarly, for the CONTROL UNIT's EEPROMs, in the WORST case, the OUTPUTS of ALL internal LOGIC GATES (or equivalent) may become FLOATING somewhere between 0 & 1, causing the FINAL OUTPUTS to become FLOATING (at 0, 1 or somewhere between 0 & 1, or oscillating among 0, 1 and somewhere between 0 & 1).

So, the ONLY solution to this problem is to CLEAR all of the FLIP-FLOPS as soon as POSSIBLE after the computer is powered up.

In REAL-WORLD computers, IMMEDIATELY AFTER the computer is powered up, a POWER-ON-RESET GENERATOR starts sending RESET signals to CLEAR all of the FLIP-FLOPS. After some time when the power STABILIZES, the initial PROPAGATIONS get completed and all of the FLIP-FLOPS get cleared, the POWER-ON-RESET GENERATOR stops sending the RESET signals and starts the MASTER CLOCK.

^ Startup Behaviour

AFTER the computer is powered up and AFTER the initial PROPAGATIONS get completed, the MASTER CLOCK becomes 0, and the RESET button is pressed, which CLEARS ALL of the FLIP-FLOPS, making their OUTPUTS 0's.
Hence, for every GATED D LATCH of the RAM, the output will remain ARBITRARY (0, 1, OSCILLATING between 0 & 1 or even FLOATING somewhere 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
1. Signals to PROPAGATE through the RESET circuit and then through the individual FLIP-FLOPS, CLEARING them and making their OUTPUTS 0's.
2. Signals to PROPAGATE for the CONTROL WORD corresponding to the ADDRESS 0000 000 00 (i.e. CO = 1, MI = 1 and the other signals = 0) to get generated.
3. Signals to PROPAGATE after CO & MI become 1 and the other signals become 0.

AFTER the RESET button is released, a LONG ENOUGH time passes for the CLEAR inputs of the individual FLIP-FLOPS to become DISABLED after signals getting PROPAGATED through the RESET circuit.

For the PROGRAM COUNTER, AFTER its FLIP-FLOPS become CLEARED and their CLEAR inputs become DISABLED, the CP inputs of all MASTER/SLAVE gated latches stay at 1/0.
But, as its COUNT input has already been at 0 since the time during which the RESET button was pressed, all MASTER & SLAVE gated latches remain in the CLEAR state, making their OUTPUTS remain 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.

AFTER the RESET button is released, the program to be run is MANUALLY programmed into the RAM by SELECTING the MANUAL mode using the PUSH BUTTON SWITCH, SETTING the ADDRESSES and the DATA using the DIP SWITCHES, and WRITING the DATA using the PUSH BUTTON. A LONG ENOUGH time passes for signals to PROPAGATE BEFORE the PUSH BUTTON used for WRITING is pressed. Similarly, the time DURATION for which the PUSH BUTTON is pressed is LONG ENOUGH for signals to PROPAGATE.

AFTER the program to be run is MANUALLY programmed into the RAM, the RUN mode is SELECTED by using the PUSH BUTTON SWITCH.