Lecture 4: Sequential Circuits

Term Test Date

- Term Test
  - July 3
  - 17:00-19:00
  - SW309
- Assembly Test
  - In class
  - End of July
  - Details TBA

B58 asks the big questions

- How does Tickle Me Elmo work?

Something else to consider…

- Computer specs use terms like "8 GB of RAM" and "2.2GHz processors".
  - What do these terms mean?
    - RAM = Random Access Memory; 8GB = 8 billion `ints`
    - 2.2 GHz = 2.2 billion clock pulses per second.
  - But what does this mean in circuitry?
    - How do you use circuits to store values?
    - What is the purpose of a clock signal?

Two kinds of circuits

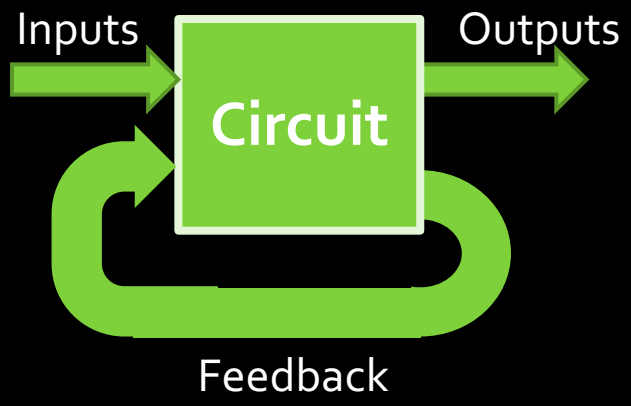
- So far, we've dealt with combinational circuits:
  - Circuits where the output values are entirely dependent and predictable from the input values.
- Another class of circuits: sequential circuits
  - Circuits that also depend on both the inputs and the previous state of the circuit.

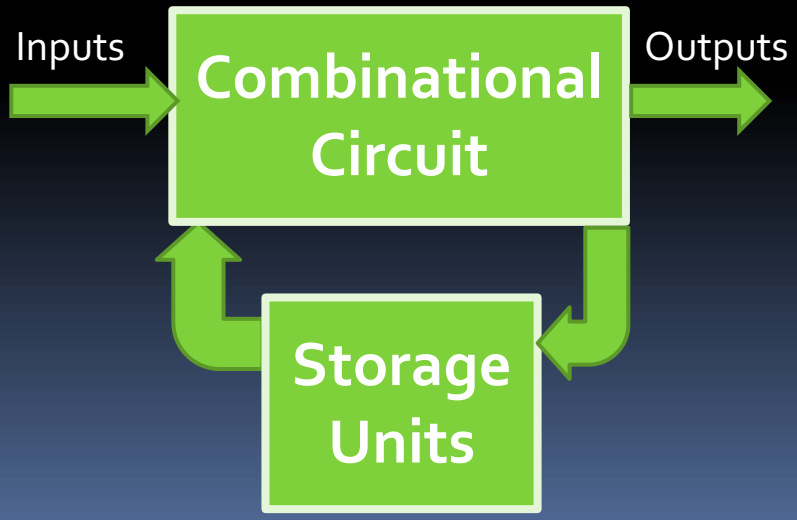
Sequential circuits

- This creates circuits whose internal state can change over time, where the same input values can result in different outputs.
- Why would we need circuits like this?
  - Memory values
  - Reacting to changing inputs

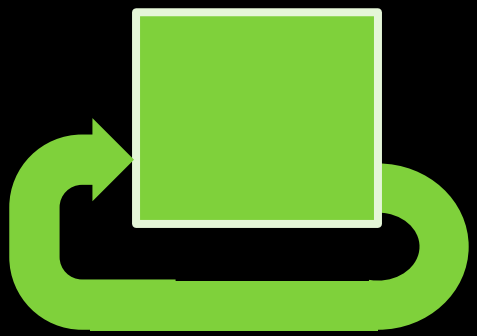
Creating sequential circuits

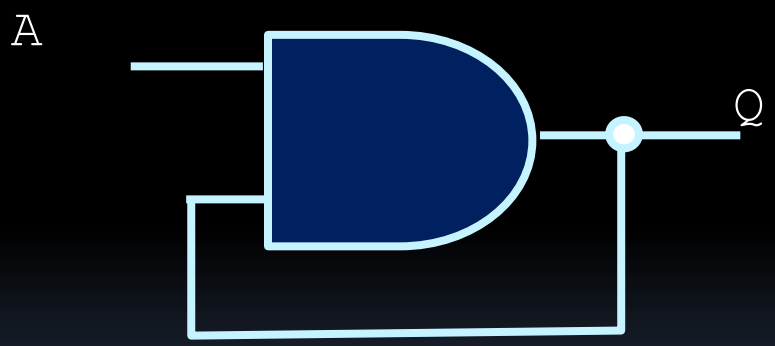
- Essentially, sequential circuits are a result of having feedback in the circuit.
  - How is this accomplished?
  - What is the result of having the output of a component or circuit be connected to its input?





Feedback



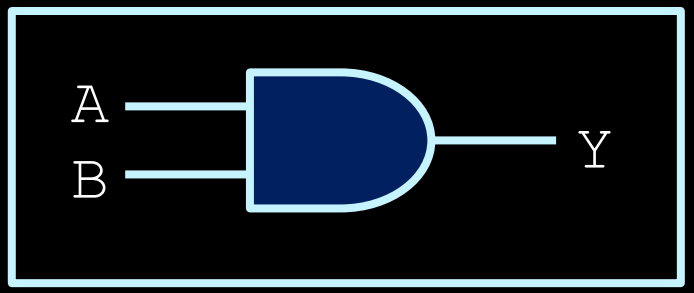


Gate Delay

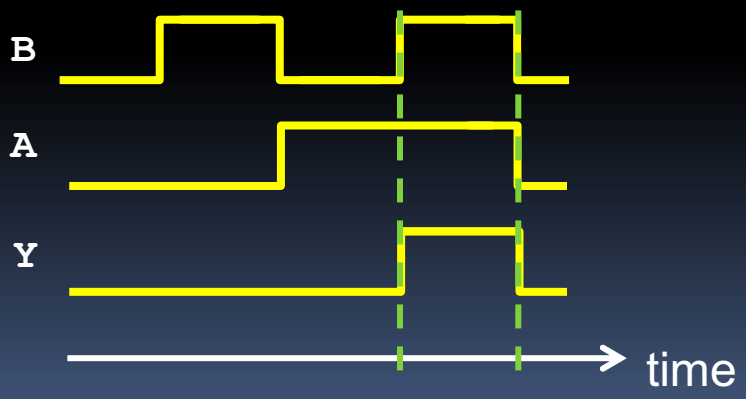
- Outputs don't change instantaneously.
  - Electrons have to move, transistors open/close…
  - Even in combinatorial circuits.

- Gate Delay or Propagation Delay:
  - "The length of time it takes for an input change to result in the corresponding output change."

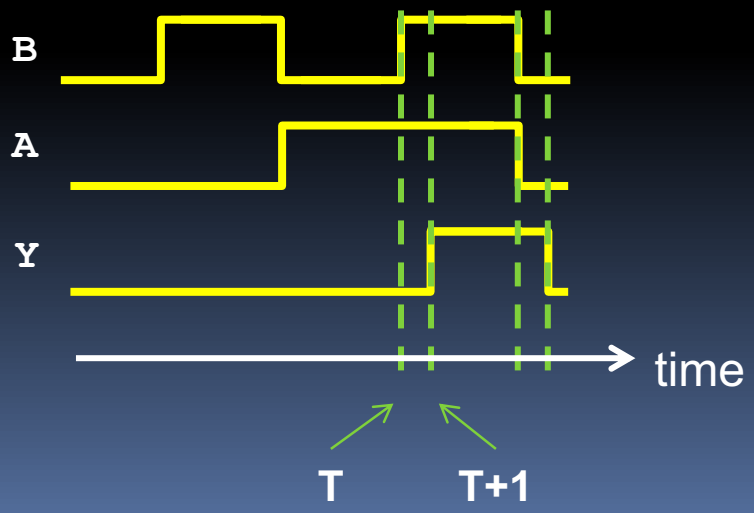
Gate Delay Example



Ideal

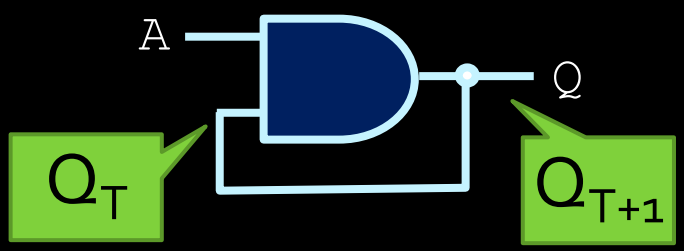


Considering delays



Feedback Circuit Example (AND)

- Some gates don't have useful results when outputs are fed back on inputs.



Q_T and Q_{T+1} represent the values of Q at a time T, and a point in time immediately after (T+1)

A	Q_T	Q_{T+1}
0	0	0
0	1	0
1	0	0
1	1	1

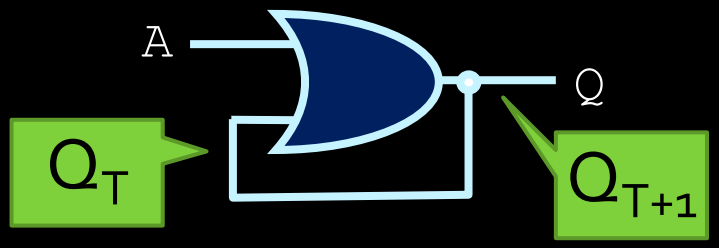
If A=0, Q_{T+1} becomes 0 no matter what Q_T was.

What happens next for later values of A?

Q_{T+1} gets stuck at 0 and cannot change ☹

Feedback Circuit Example (OR)

- Some gates don't have useful results when outputs are fed back on inputs.



In this truth table, Q_T and Q_{T+1} represent the values of Q at a time T, and a point in time immediately after (T+1)

A	Q_T	Q_{T+1}
0	0	0
0	1	1
1	0	1
1	1	1

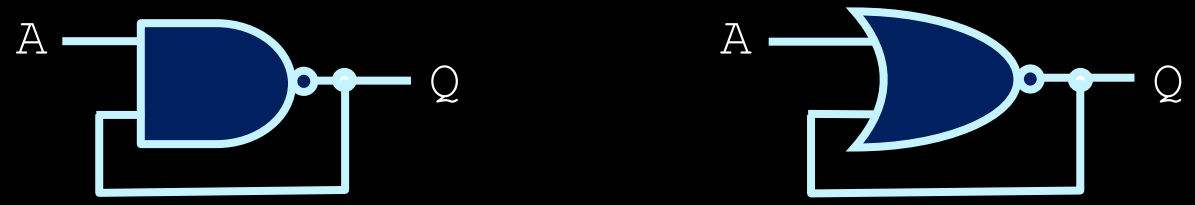
If A=1, Q_{T+1} becomes 1 no matter what Q_T was.

What happens next for later values of A?

Q_{T+1} gets stuck at 1. Not very useful ☹

Feedback Examples (NAND, NOR)

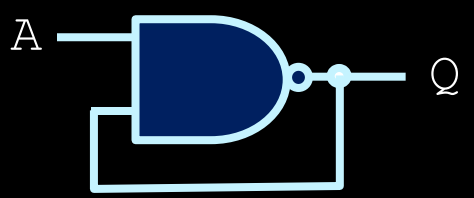
- NAND, NOR gates w/ feedback have more interesting characteristics, which lend themselves to storage devices.



- What makes NAND and NOR feedback circuits different?
  - Unlike the AND and OR gate circuits (which get stuck), the output Q_{T+1} can be changed, based on A.

Feedback Example (NAND)

- Let's assume we set A=0
  - Then, output Q will go to 1.
  - If we leave A unchanged we can store 1 indefinitely!
- If we set A=1, Q's value can change, but there's a catch!

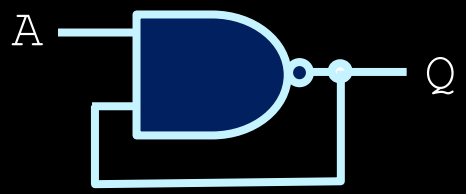


A	Q_T	Q_{T+1}
0	0	1
0	1	1
1	0	1
1	1	0

What happens in these last two scenarios?

Unsteady state! Can't store 0 long!

NAND waveform behaviour

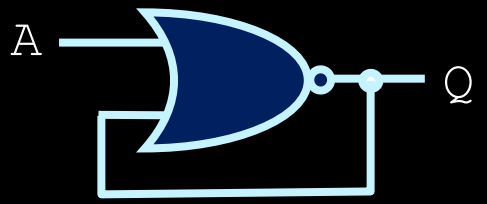


A	Q_T	Q_{T+1}
0	0	1
0	1	1
1	0	1
1	1	0

Gate delay. Output does not change instantaneously

Feedback Example (NOR)

- Let's assume we set A=1
- Then, output Q will go to 0.
- If we leave A unchanged we can store 0 indefinitely!
- If we flip A, we can change Q, but there's a catch here too!



A	Q_T	Q_{T+1}
0	0	1
0	1	0
1	0	0
1	1	0

Feedback behaviour

- NAND behaviour

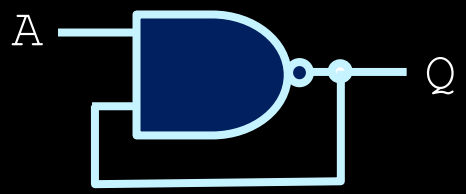
A	Q_T	Q_{T+1}
0	0	1
0	1	1
1	0	1
1	1	0

- NOR behaviour

A	Q_T	Q_{T+1}
0	0	1
0	1	0
1	0	0
1	1	0

- Output Q_{T+1} can be changed, based on A.
- However, gates like these that feed back on themselves could enter an unsteady state.

NAND waveform behaviour



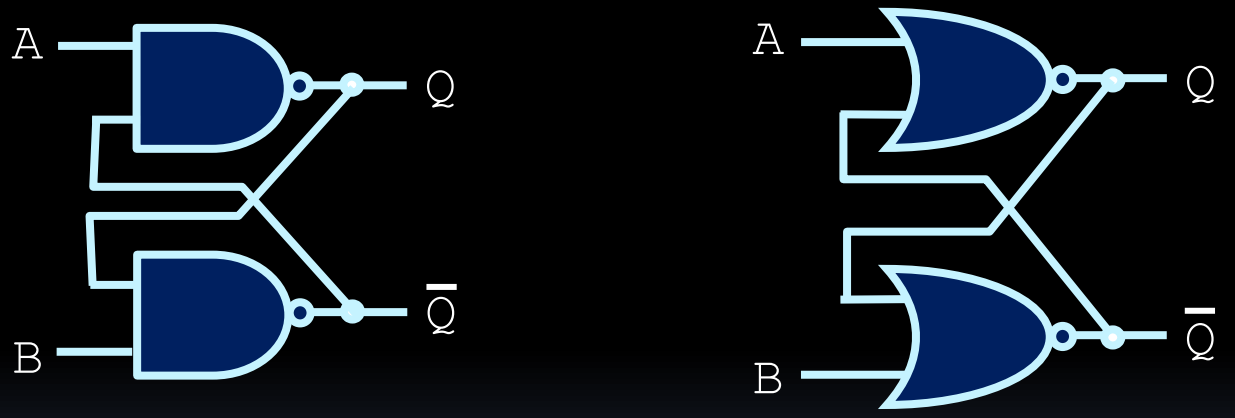
A	Q_T	Q_{T+1}
0	0	1
0	1	1
1	0	1
1	1	0

Gate delay. Output does not change instantaneously

We want to avoid this. We should be able to store high and low values for as long as we want, and change those values as needed.

Latches

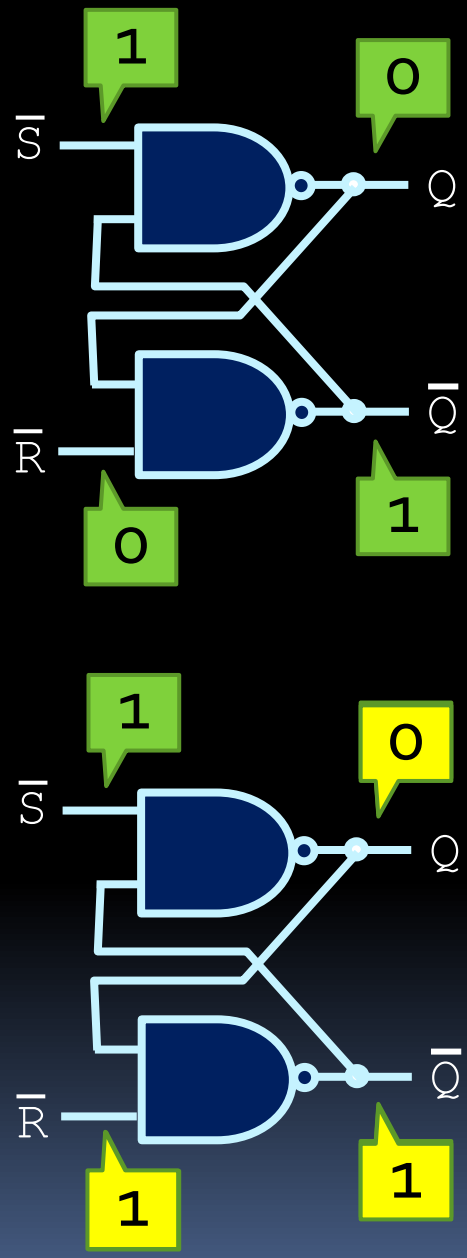
- If multiple gates of these types are combined, you can get more steady behaviour.



- These circuits are called latches.

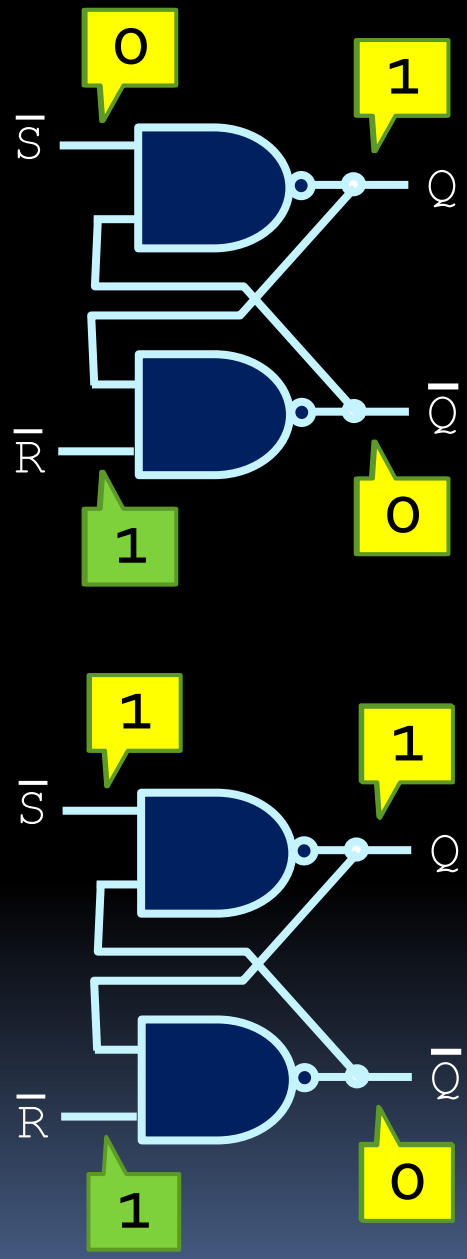
$\overline{S}\overline{R}$ latch

- Let's see what happens when the input values are changed…
  - Assume that $\overline{S}$ and $\overline{R}$ are set to 1 and 0 to start.
  - The $\overline{R}$ input sets the output $\overline{Q}$ to 1, which sets the output Q to 0.
  - Setting $\overline{R}$ to 1 keeps the output value $\overline{Q}$ at 1, which maintains both output values.

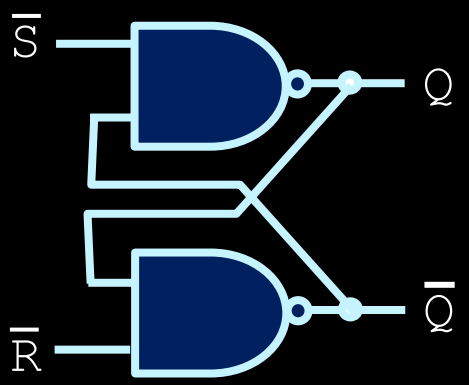


$\overline{S}\overline{R}$ latch

- (continuing from previous)
  - $\overline{S}$ and $\overline{R}$ start with values of 1, when $\overline{S}$ is set to 0.
  - This sets output Q to 1, which sets the output $\overline{Q}$ to 0.
  - Setting $\overline{S}$ back to 1 keeps the output value $\overline{Q}$ at 0, which maintains both output values.
- Note: inputs of 11 maintain the previous output state!



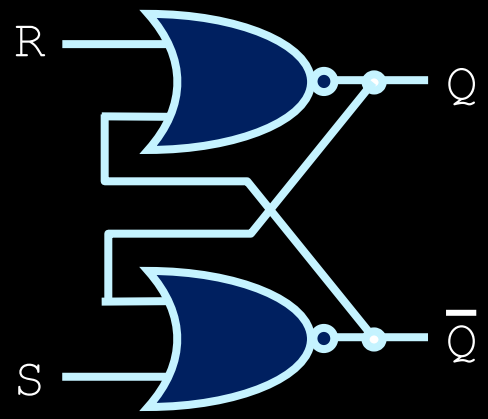
$\overline{S}\overline{R}$ latch



$\overline{S}$	$\overline{R}$	Q_T	$\overline{Q}_T$	Q_{T+1}	$\overline{Q}_{T+1}$
0	0	X	X	1	1
0	1	X	X	1	0
1	0	X	X	0	1
1	1	0	1	0	1
1	1	1	0	1	0

- $\overline{S}$ and $\overline{R}$ are called "set" and "reset" respectively.
- Note how the circuit "remembers" its signal when going from 10 or 01 to 11.
- Going from 00 to 11 produces unstable behaviour!
  - Depending on which input changes first.

SR latch

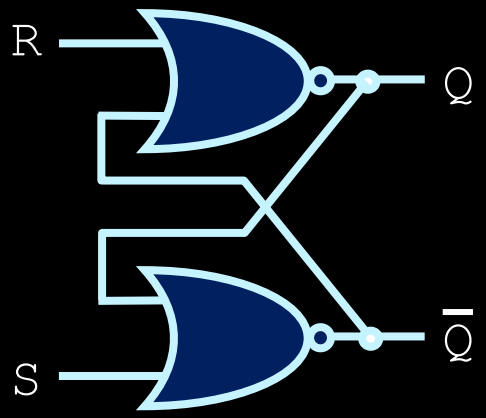


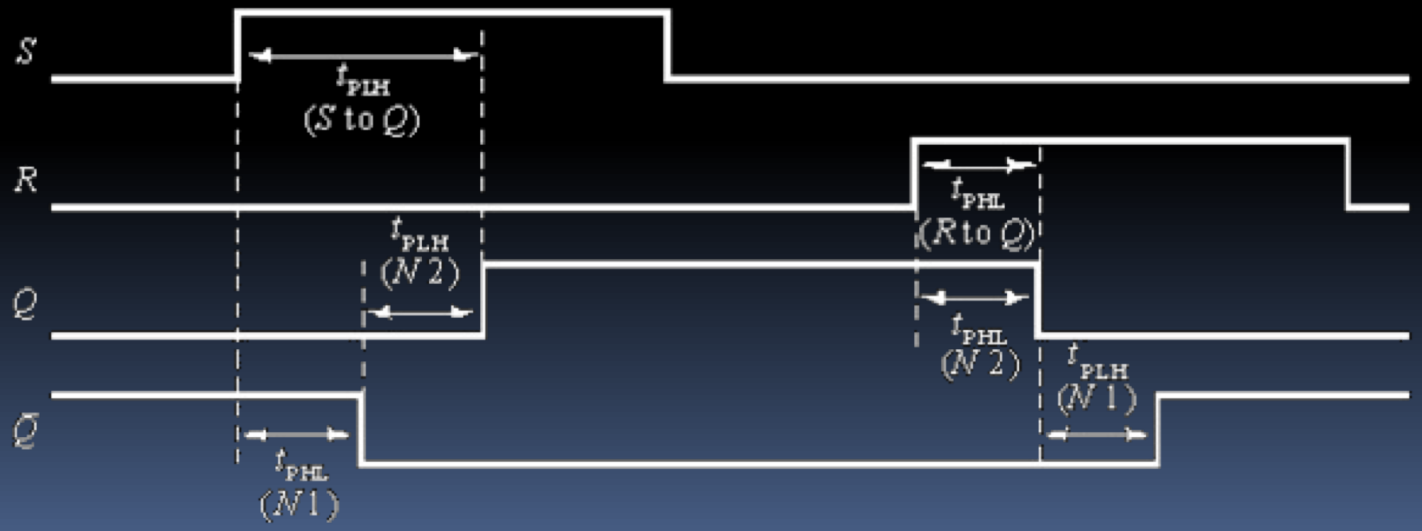
S	R	Q_T	$\overline{Q}_T$	Q_{T+1}	$\overline{Q}_{T+1}$
0	0	0	1	0	1
0	0	1	0	1	0
0	1	X	X	0	1
1	0	X	X	1	0
1	1	X	X	0	0

- In this case, S and R are "set" and "reset".
- In this case, the circuit "remembers" previous output when going from 10 or 01 to 00.
- As with $\overline{S}\overline{R}$ latch, unstable behaviour is possible, but this time when inputs go from 11 to 00.

SR latch timing diagram

- Important to note that the output signals don't change instantaneously.





More on instability

- Unstable behaviour occurs when a $\overline{S}\overline{R}$ latch goes from `00` to `11`, or a SR latch goes from `11` to `00`.
  - The signals don't change simultaneously, so the outcome depends on which signal changes first.
- Because of the unstable behaviour, `00` is considered a forbidden state in NAND-based $\overline{S}\overline{R}$ latches, and `11` is considered a forbidden state in NOR-based SR latches.

Reading from latches

- Now we have circuit units that can store high or low values. How can we read from them?
  - For instance, when do we know when the output is ready to be sampled?
  - If the output is high, how can we tell the difference between a single high value and two high values in a row?
- Need some sort of timing signal, to let the circuit know when the output may be sampled.

  → clock signals.