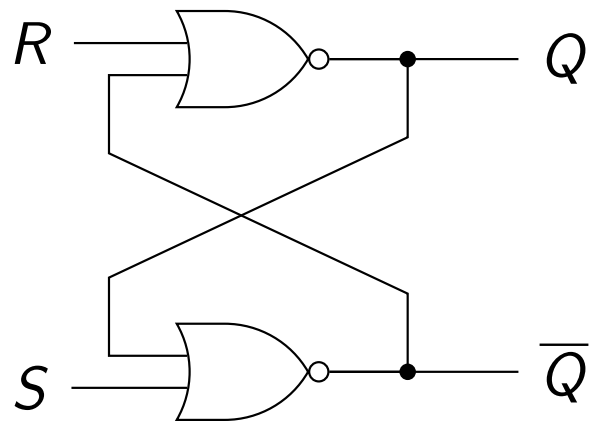
Latches

- One type of *storage element* — exhibits the concept of memory.

SR latch

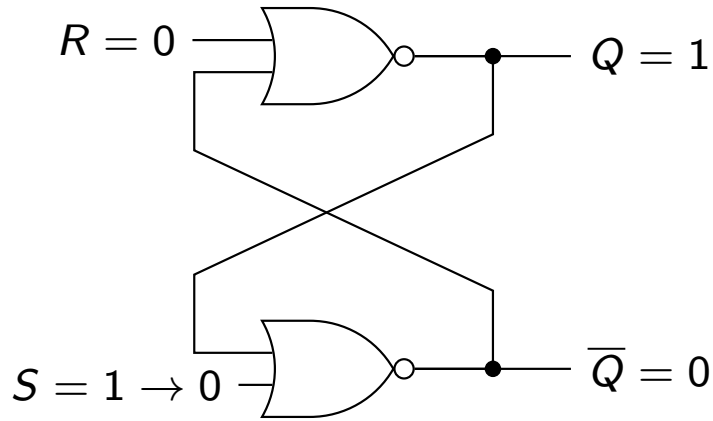
- We want a circuit in which we can force one output to 1, force the output to 0 and "hold" the output. Further, the circuit will have 2 outputs which are always *complements* of each other.
- The SR latch:



- There are 3 different cases to consider to see if this circuit does what we want and to figure out how it does it.

SR latch

- Case I: Set $S = 1$ and $R = 0$. Then, change S to $1 \rightarrow 0$.
- The *SR* latch:

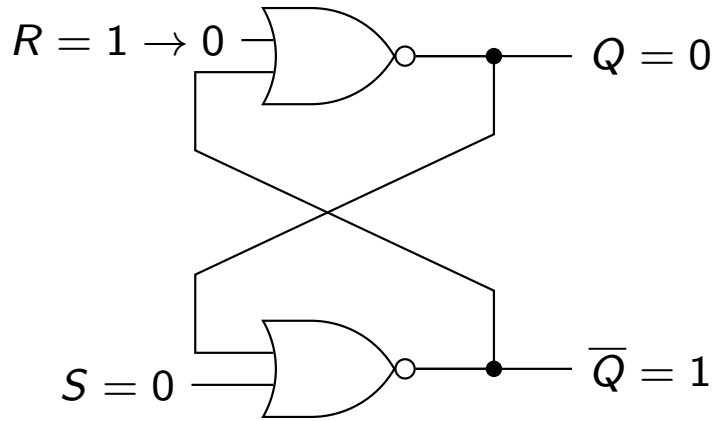


- Behaviour: By considering how a **NOR** gate works, setting $S = 1$ causes the bottom **NOR** to produce $\overline{Q} = 0$. The top **NOR** gate has inputs 00 and therefore produces $Q = 1$. We will call this the *set* state.
- Next, S is changed to 0. The bottom **NOR** gate now has inputs 10 which means that $\overline{Q} = 0$ still. The top **NOR** gate remains unchanged and $Q = 1$. We will call this the *memory* or *hold* state.
- Summary:

$$\begin{array}{cccccc} S = 1 & R = 0 & \rightarrow & Q = 1 & \overline{Q} = 0 & \leftarrow \textit{set} \\ & & \downarrow & & & \\ S = 0 & R = 0 & \rightarrow & Q = 1 & \overline{Q} = 0 & \leftarrow \textit{hold} \end{array}$$

SR latch

- Case II: Set $S = 0$ and $R = 1$. Then, change R to $1 \rightarrow 0$.
- The *SR* latch:

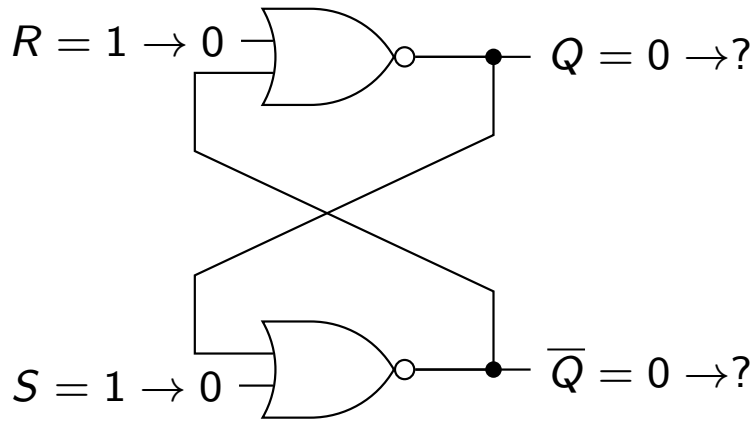


- Behaviour: By considering how a **NOR** gate works, setting $R = 1$ causes the top **NOR** to produce $Q = 0$. The bottom **NOR** gate has inputs 00 and therefore produces $\overline{Q} = 1$. We will call this the *reset* state.
- Next, R is changed to 0. The top **NOR** gate now has inputs 01 which means that $Q = 0$ still. The bottom **NOR** gate remains unchanged and $\overline{Q} = 1$. We will call this the *memory* or *hold* state.
- Summary:

$$\begin{array}{cccccc} S = 0 & R = 1 & \rightarrow & Q = 0 & \overline{Q} = 1 & \leftarrow \textit{reset} \\ & & \downarrow & & & \\ S = 0 & R = 0 & \rightarrow & Q = 0 & \overline{Q} = 1 & \leftarrow \textit{hold} \end{array}$$

SR latch

- Case III: Set $S = 1$ and $R = 1$.
- The *SR* latch:



- With both $S = 1$ and $R = 1$ we see that both $Q = 0$ and $\overline{Q} = 0$ which should be troubling — the outputs are no longer complements of each other.
- Now assume that both S and R change to 0. Different things could happen...
- We will call the situation with $S = 1$ and $R = 1$ the *restricted* or *undesireable* state.

SR latch

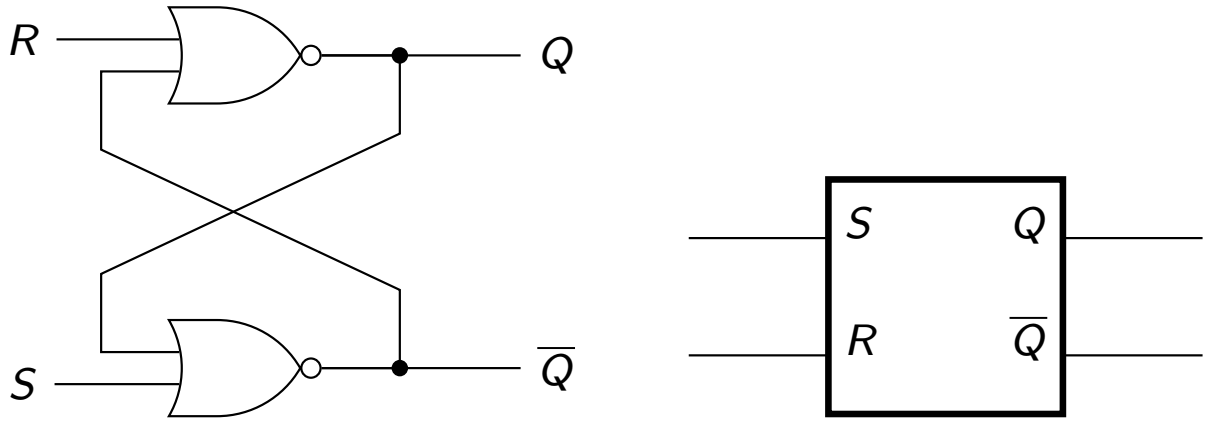
- Case III: Change $S = 1 \rightarrow 0$ and $R = 1 \rightarrow 0$.
- Both **NOR** gates have inputs 00 which means their outputs need to change... But we can assume things change at different times...
- Assume that Q changes first from $Q = 0 \rightarrow 1$.
  - This means the bottom **NOR** gate will inputs 10 and so $\overline{Q} = 0$.
  - In this case, the different circuit values are $S = 0$, $R = 0$ and $Q = 1$ and $\overline{Q} = 0$.
- Assume that $\overline{Q}$ changes first from $\overline{Q} = 0 \rightarrow 1$.
  - This means the top **NOR** gate will inputs 01 and so $Q = 0$.
  - In this case, the different circuit values are $S = 0$, $R = 0$ and $Q = 0$ and $\overline{Q} = 1$.
- Assume that both Q and $\overline{Q}$ changes at the same time from $Q = 0 \rightarrow 1$ and $\overline{Q} = 0 \rightarrow 1$.
  - But this means that both **NOR** gates will now have inputs 01 so there outputs should change back to $Q = 0$ and $\overline{Q} = 0$.
  - This situation could cause *oscillations*.
- The bottom line is that $S = 1$ and $R = 1$ is an input combination that should be avoided.

SR latch

- We can avoid $S = 1$ and $R = 1$ by adding additional logic at the inputs such that we never get this input.
- If we add logic such that $S = 1$ and $R = 1$ causes the outputs to become $Q = 1$ and $\overline{Q} = 0$, we have a *set dominated latch*.
- If we add logic such that $S = 1$ and $R = 1$ causes the outputs to become $Q = 0$ and $\overline{Q} = 1$, we have a *reset dominated latch*.
- We could do something else like have $S = 1$ and $R = 1$ cause the latch to *flip* or *toggle* its outputs. In this case we will get a *JK latch*.

SR latch

- Summary:
- The SR latch:



- Tabular description:

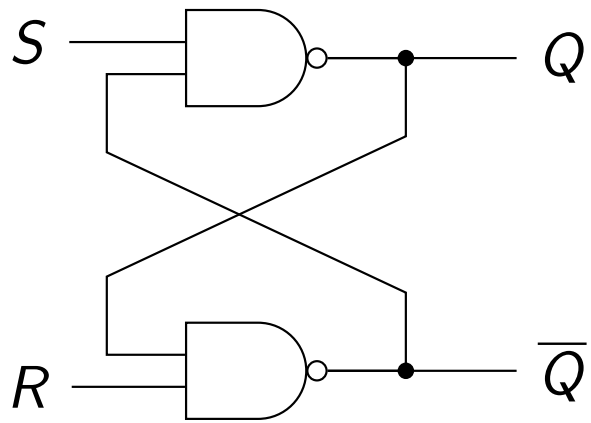
S	R	Q	$\overline{Q}$	**Action**
1	0	1	0	*set*
0	1	0	1	*reset*
0	0	1	0	*memory or hold* after $S = 1$, $R = 0$
0	0	0	1	*memory or hold* after $S = 0$, $R = 1$
1	1	?	?	*restricted*

- As an equation ($Q(t+1)$ is the "next value of Q and $Q(t)$ is the current value of Q):

$$Q(t+1) = Q(t)\overline{R} + S\overline{R}$$

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

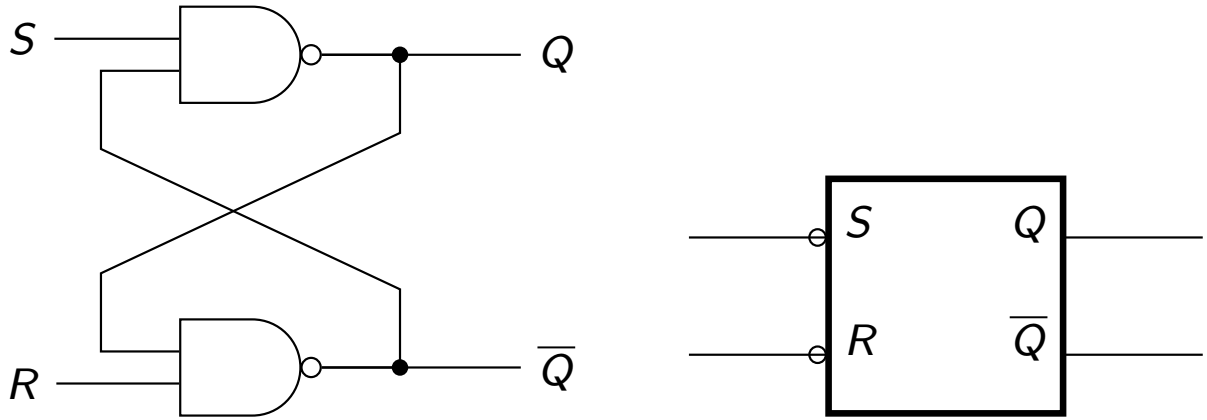
- Can build a similar circuit using **NAND** gates, but the operation is a little bit different.
- The $\overline{S}\,\overline{R}$ latch:



- We could do analysis similar to the SR latch and conclude there is a *set*, *reset*, *hold* and *restricted* case.

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

- Summary:
- The $\overline{S}\,\overline{R}$ latch:



- Tabular description:

S	R	Q	$\overline{Q}$	**Action**
1	0	0	1	*reset*
0	1	1	0	*set*
1	1	0	1	*memory or hold* after $S = 1$, $R = 0$
1	1	1	0	*memory or hold* after $S = 0$, $R = 1$
0	0	?	?	*restricted*

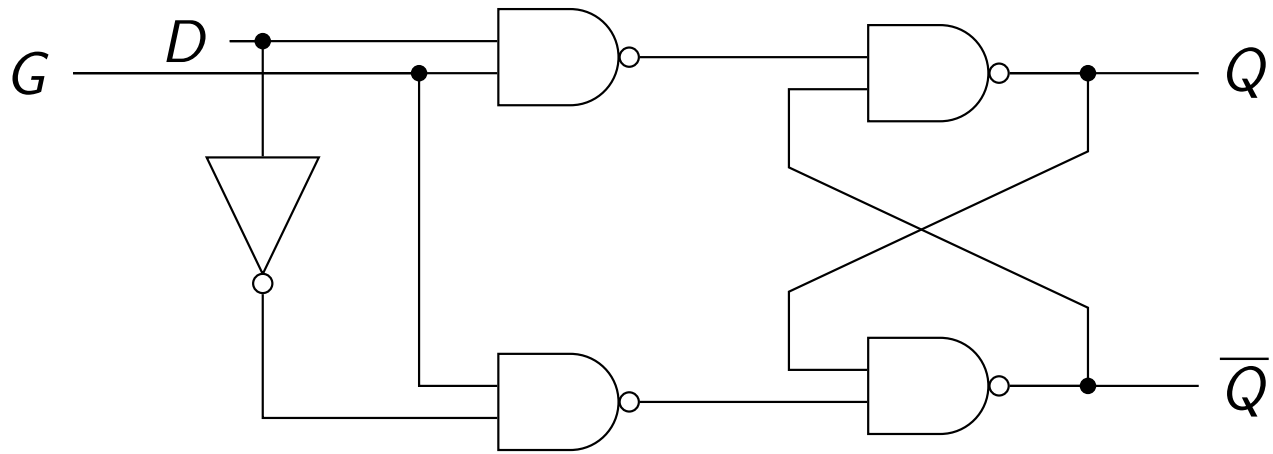
- As an equation ($Q(t+1)$ is the "next value of Q and $Q(t)$ is the current value of Q):

$$Q(t+1) = Q(t)R + \overline{S}R$$

- Note the bubbles on the S and R inputs; this tends to be described as "active low" inputs.

Gated D latch

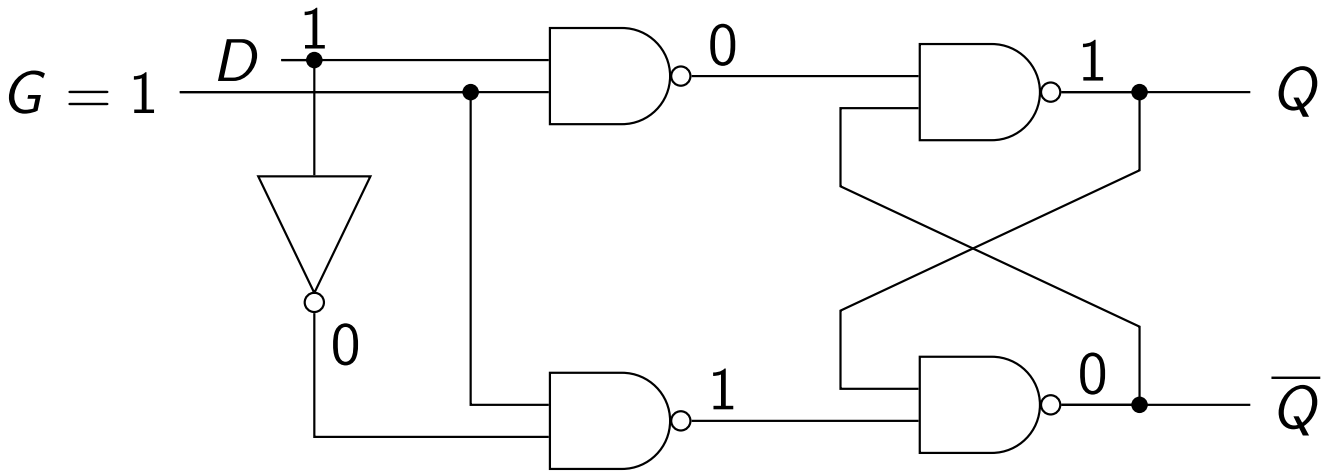
- The gated D latch:



- The D input is called the *data* input and the G input is called the *gate* input.

Gated D latch

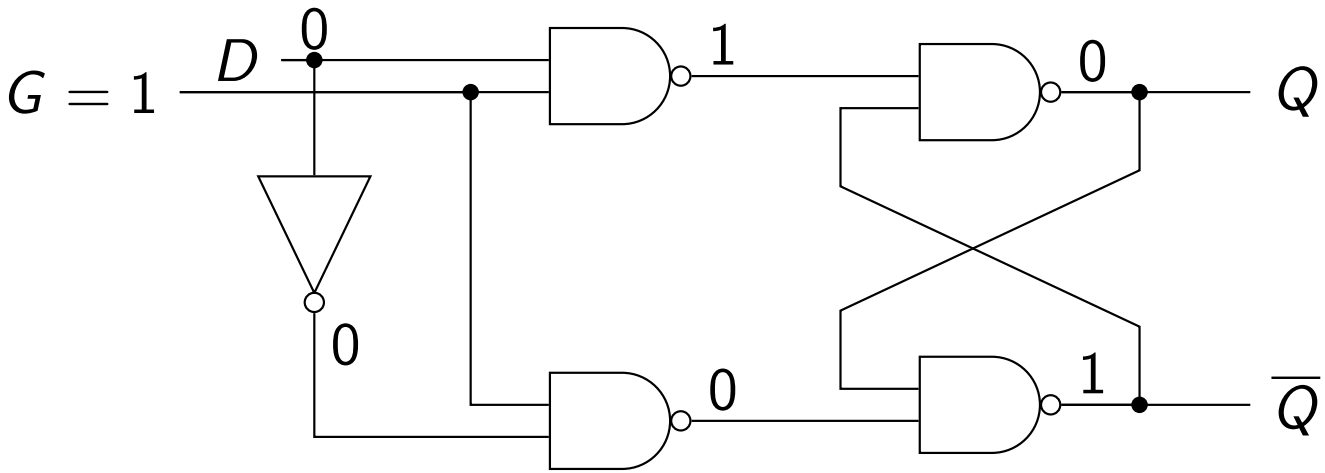
- Case I:



- If $G = 1$ and $D = 1$, then the output is *set*.

Gated D latch

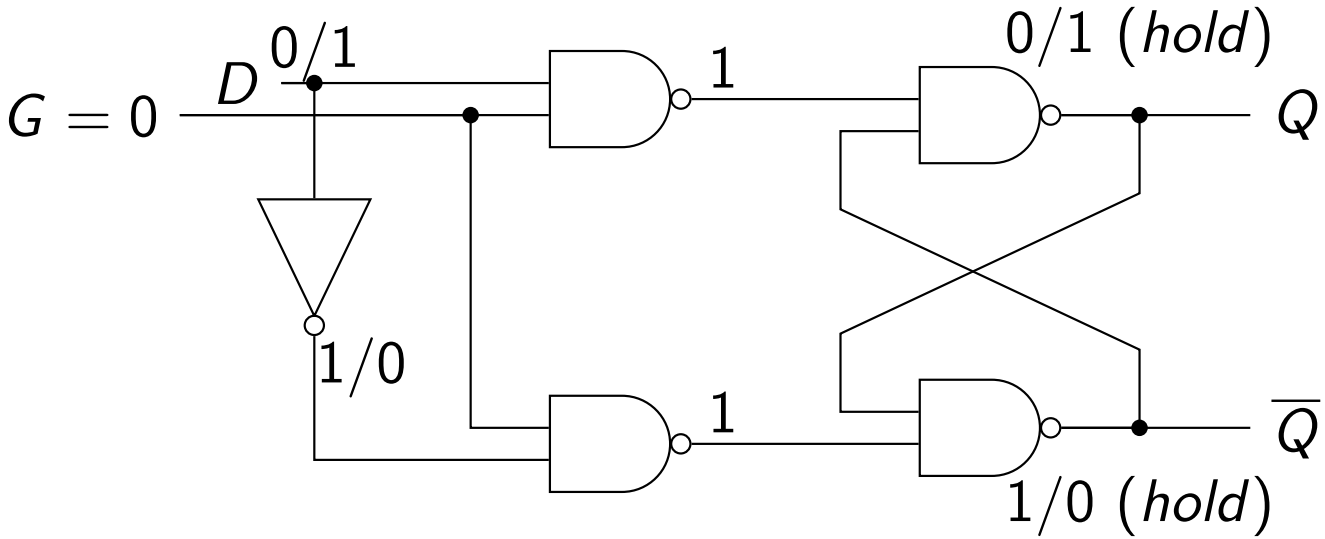
- Case II:



- If $G = 1$ and $D = 0$, then the output is *reset*.

Gated D latch

- Case III:



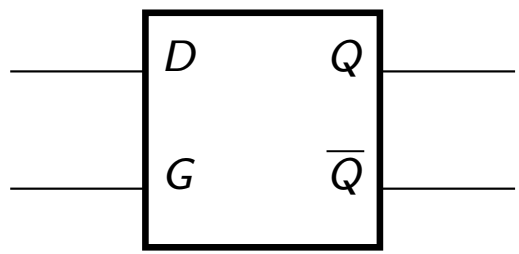
- If $G = 0$ the value of D is not relevant; the outputs remain at their previous value and *hold*.

Gated D latch

- Behaviour of the D latch:

D	G	Q	$\overline{Q}$	**Action**
1	1	1	0	$\leftarrow$ *set*
0	1	0	1	$\leftarrow$ *reset*
0/1	0	*previous value*		$\leftarrow$ *hold*

- Note that we don't need to worry about any "*restricted*" cases like with the SR latch or the $\overline{S}\,\overline{R}$ latch.
- Symbol:



- As an equation ($Q(t+1)$ is the "next value of Q and $Q(t)$ is the current value of Q):

$$Q(t+1) = DG + Q(t)\overline{G}$$

Latch summary

- There are other configurations and types of latches, but these ones are enough for our purposes.
- We won't do too much with latches depending on time, but they are useful for a number of things...
- One last comment... Latches are an example of a *level sensitive* storage element — the outputs are *set*, *reset* or *hold* depending on the logic levels/values of the inputs.