EE103 Digital Systems 1

Wong Chin Hong

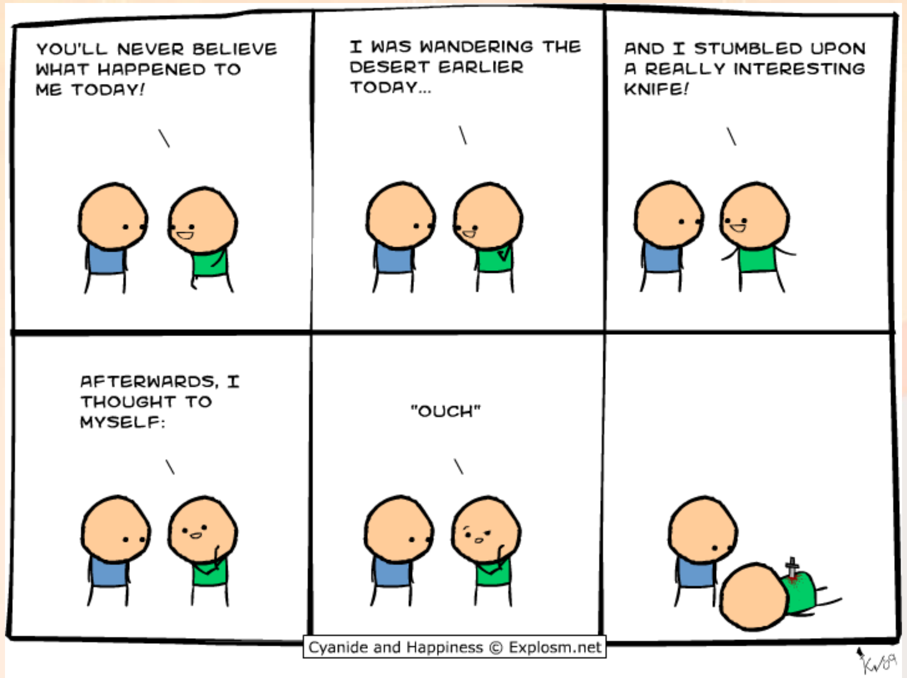
106

Maynooth International Engineering College
Fuzhou University

So far … !

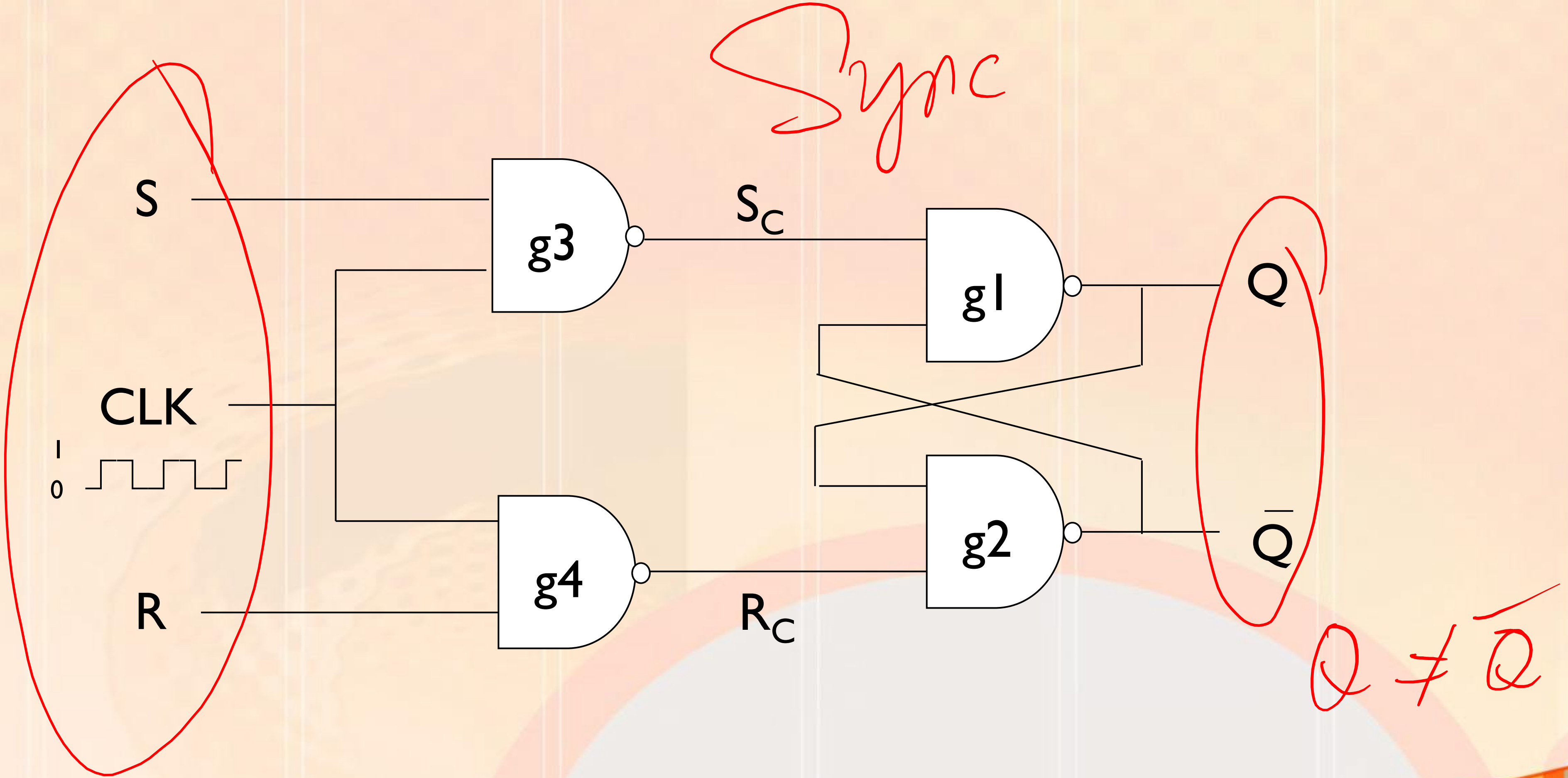
- We've used Karnaugh Maps to minimise logic ...
- We've implemented circuits using NAND only or NOR only gates ...
- We've introduced Sequential Logic and looked in detail at both the asynchronous and the synchronous SR flipflop …

TODAY, we are going to look at the D, JK and T flipflops ...



***BUT FIRST**, we are going to finish off the section on the SR flipflop...*

Synchronous SR Flipflop



Synchronous SR Flipflop

- By carefully analysing the state table, we can derive the following requirements table for the SR flipflop, where X represents don't care terms:

Sync SR ff

S	R	Q	Q(τ)
0	0	0	**0**
0	0	1	**1**
0	1	0	**0**
0	1	1	**0**
1	0	0	**1**
1	0	1	**1**
1	1	0	**-**
1	1	1	**-**

}Q

}0

}1

} undefine.

Operation	**S**	**R**
Stay at 0	**0**	**X**
Stay at 1	**X**	**0**
Go to 0	**0**	**1**
Go to 1	**1**	**0**

Synchronous SR Flipflop

- We can also express the flipflop behaviour in terms of a **Next State Equation**.

- This is simply a Boolean expression that relates the next output $Q(\tau)$ to the inputs S and R and the previous output Q.

- The next state equation can be derived in two ways …

Synchronous SR Flipflop

Karnaugh Map …

S	R	Q	Q(τ)	
0	0	0	0	0
0	0	1	1	1
0	1	0	0	2
0	1	1	0	3
1	0	0	1	4
1	0	1	1	5
1	1	0	~~-~~	6
1	1	1	~~-~~	

Can't happen – assign as don't care terms

X

SR

Q				
	0 (0)	0 (2)	X (6)	1 (4)
	1 (1)	0 (3)	X (7)	1 (5)

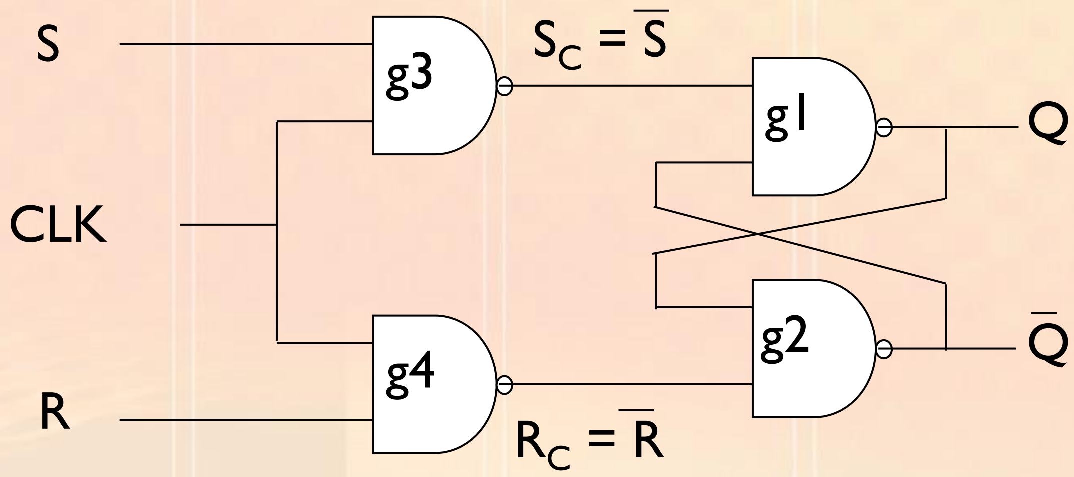
$$\boxed{Q(\tau) = R + \bar{S}Q}$$

Synchronous SR Flipflop

$\boxed{\text{Logic Circuit …}}$

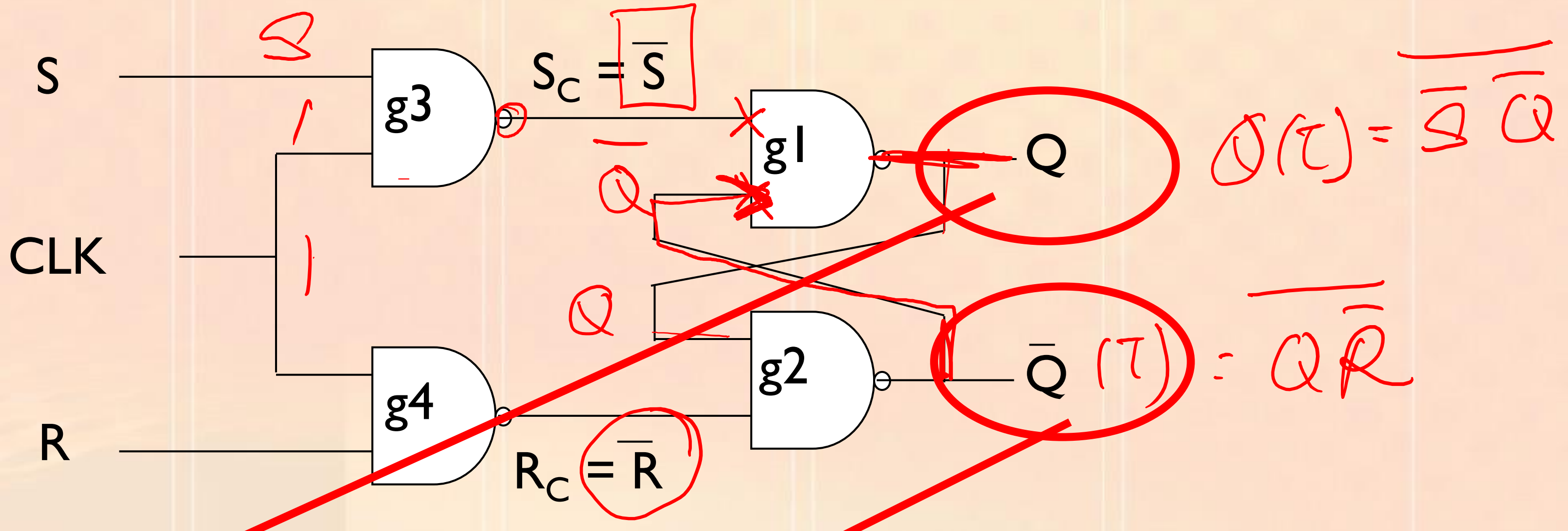
$\cancel{Q(T) = R + \bar{S}Q}$

$Q(T) = S + \bar{R}Q$



Synchronous SR Flipflop

Logic Circuit …

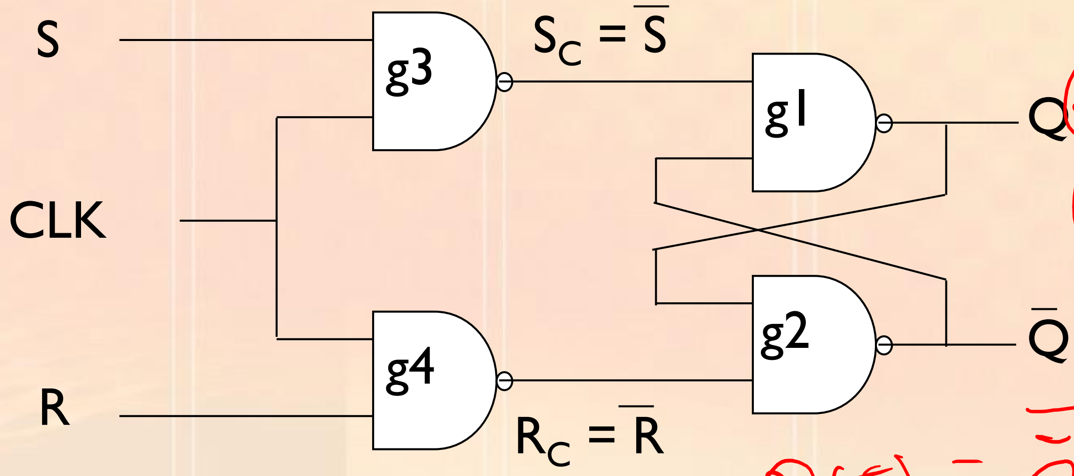


$$Q(\tau) = \overline{\overline{S}\;\overline{Q}}$$

$$\overline{Q(\tau)} = \overline{Q\,\overline{R}}$$

Synchronous SR Flipflop

Logic Circuit …



① $\cdot \leftrightarrow +$

② bar each term

③ bar everything

④ simplify the complement

$$\boxed{Q(\tau) = S + \overline{R}Q}$$

De Morgan

$$Q(\tau) = \overline{\overline{S}\;\overline{Q}}$$

$$\overline{Q(\tau)} = \boxed{\overline{Q\,\overline{R}}}$$

$$\Rightarrow Q(\tau) = \overline{\overline{S}\;\boxed{\overline{\overline{Q\,\overline{R}}}}}$$

$$Q(\tau) = \overline{\overline{\overline{S}}\;\overline{Q\,\overline{R}}}$$

$$= \overline{\overline{S}} + \overline{\overline{Q\,\overline{R}}}$$

$$\boxed{Q(\tau) = S + Q\overline{R}}$$

$$\overline{R}Q = Q\overline{R}$$

Synchronous SR Flipflop

- Finally, we can also express the operation of flipflop in a graphical fashion known as a **state diagram**.

- The state diagram describes the possible states of the flipflop along with the conditions for how it can transition between these states.

- The state diagram for the synchronous SR flipflop is as follows:

SR	S	R
stay 0	0	X
stay 1	X	0
go→0	0	1
go→1	1	0

D (Data) Flipflop

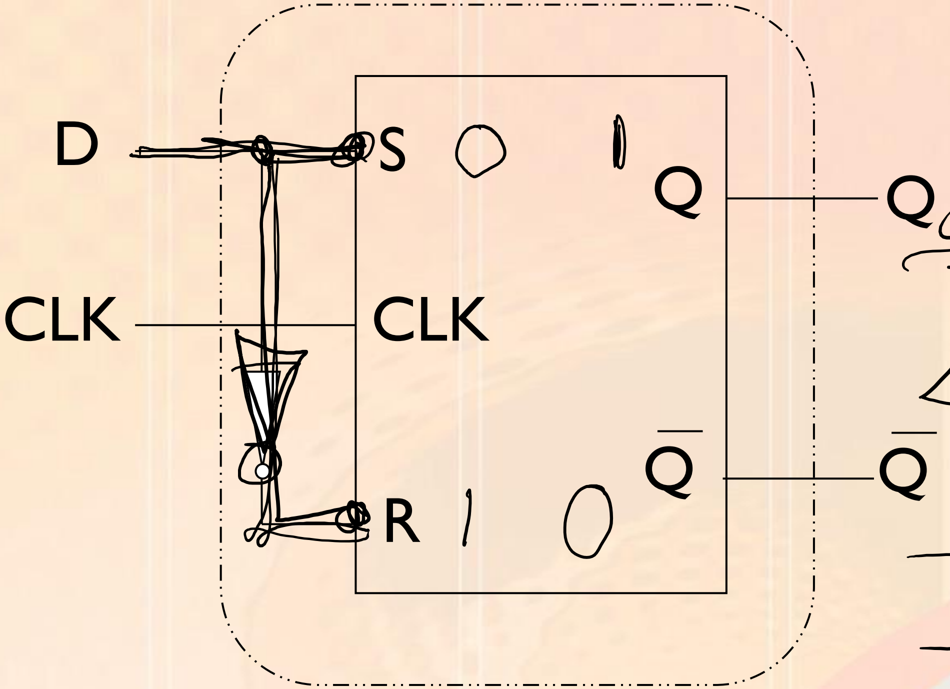
D
JK
T
S
R

- The **D or data flipflop** is the simplest of all the flipflops.
- It basically allows the transfer of data in a synchronised manner.
- The D flipflop is a modified version of the SR flipflop where the inputs S and R are tied together with an inverter between them, as follows:

D (Data) Flipflop

D=0

D=1



S ≠ R

D	S	R	Q	Q(n)	
~~X~~	~~0~~	~~0~~	~~0/1~~		~~Q~~
0	0	1	0/1	0/0	0 reset
1	1	0	0/1	1/1	1 set
~~X~~	~~1~~	~~1~~	~~0/1~~		undefine

D (Data) Flipflop

- The **state table** is easily derived from that of the SR flipflop as follows:

D	S	R	Q(τ)	
0	0	1	**0**	*Q is set to 0*
1	1	0	**1**	*Q is set to 1*

D	Q(τ)
0	**0**
1	**1**

D (Data) Flipflop

- Note that the condition S = R cannot occur due to the presence of the inverter and, hence, the indeterminate state of the SR flipflop is not an issue here.

- In brief, for the D flipflop, *the next output is simply the same as the input* (once the CLK is high).

D	Q(τ)
0	**0**
1	**1**

D (Data) Flipflop

- The **requirements table** is as follows:

Operation	D
Stay at 0	0
Stay at 1	1
Go to 0	0
Go to 1	1

1 → 0

D	S	R	Q	Q(T)
	0	0	0	Q
	0	0	1	
0	0	1	0 →	0
	0	1	1 →	0
1	1	0	0 →	1
	1	0	1 →	1
	1	1	0	–
	1	1	1	–

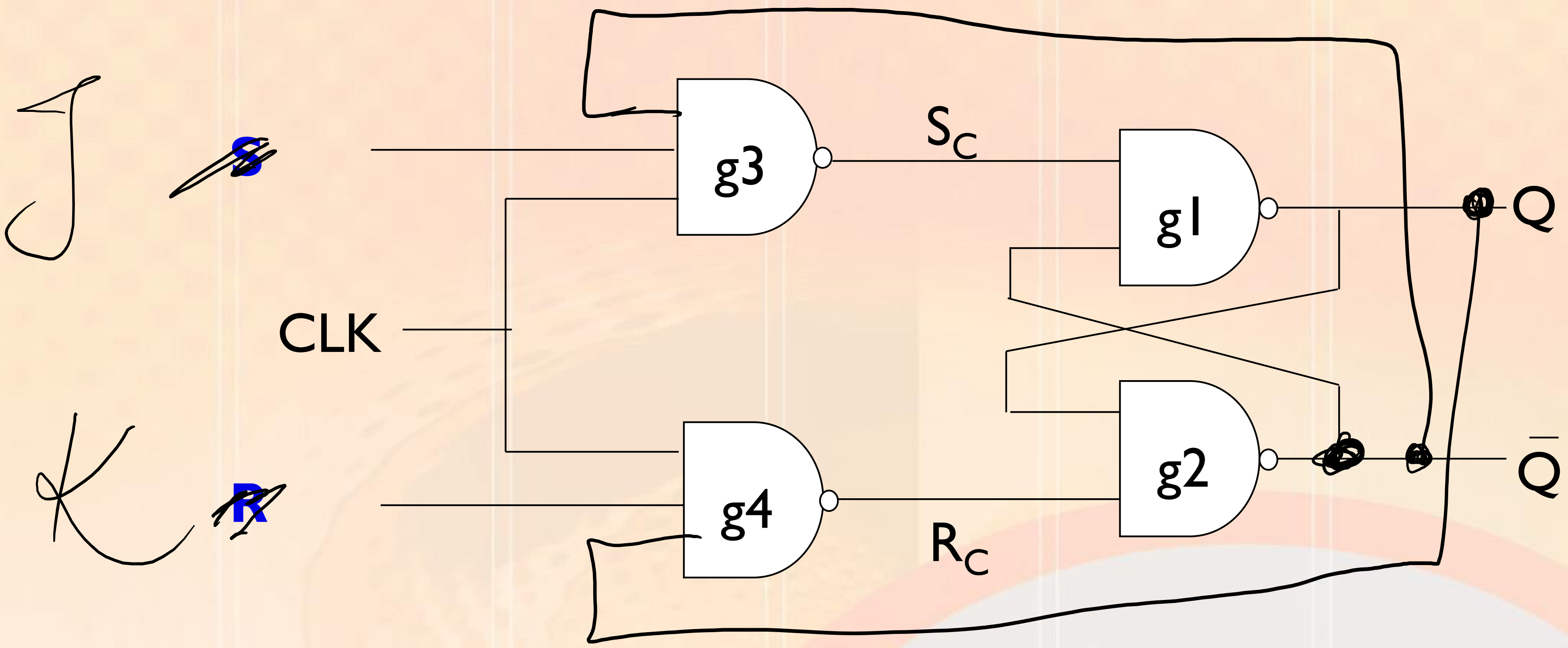
JK Flipflop

- The JK flipflop is a refined version of the SR flipflop, where the indeterminate state of the SR flipflop is now defined.

undefine

- Here, **S** now becomes **J** and **R** becomes **K**, as follows:

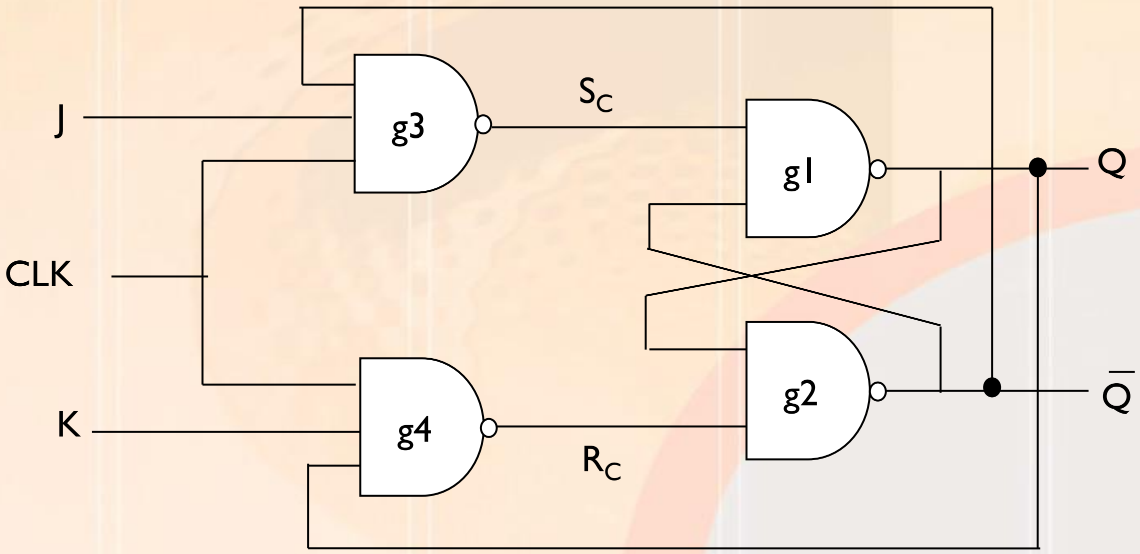
JK Flipflop



SR flipflop

JK Flipflop

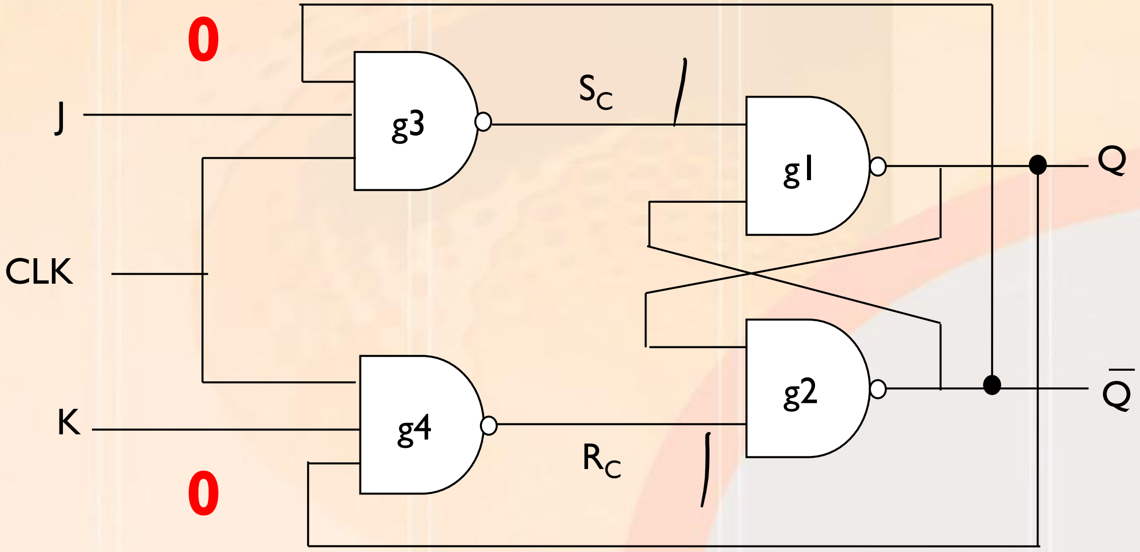
- Let us now consider the operation of the JK flipflop ...



JK Flipflop

A	B	f
0	0	1
0	1	1
1	0	1
1	1	

- If **J = 0** and **K = 0** …



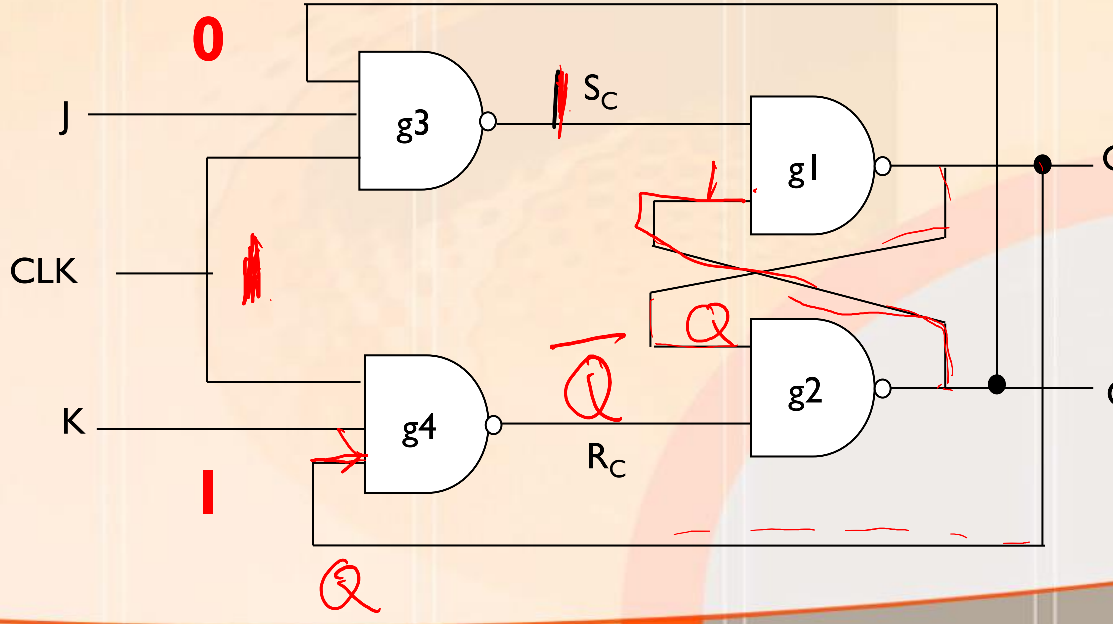
S	R	Q	Q(T)
1	1	0	0
		1	1

} Q

JK Flipflop

A	B	f
0	0	1
0	1	1
1	0	1
1	1	0

- If **J = 0** and **K = 1** …



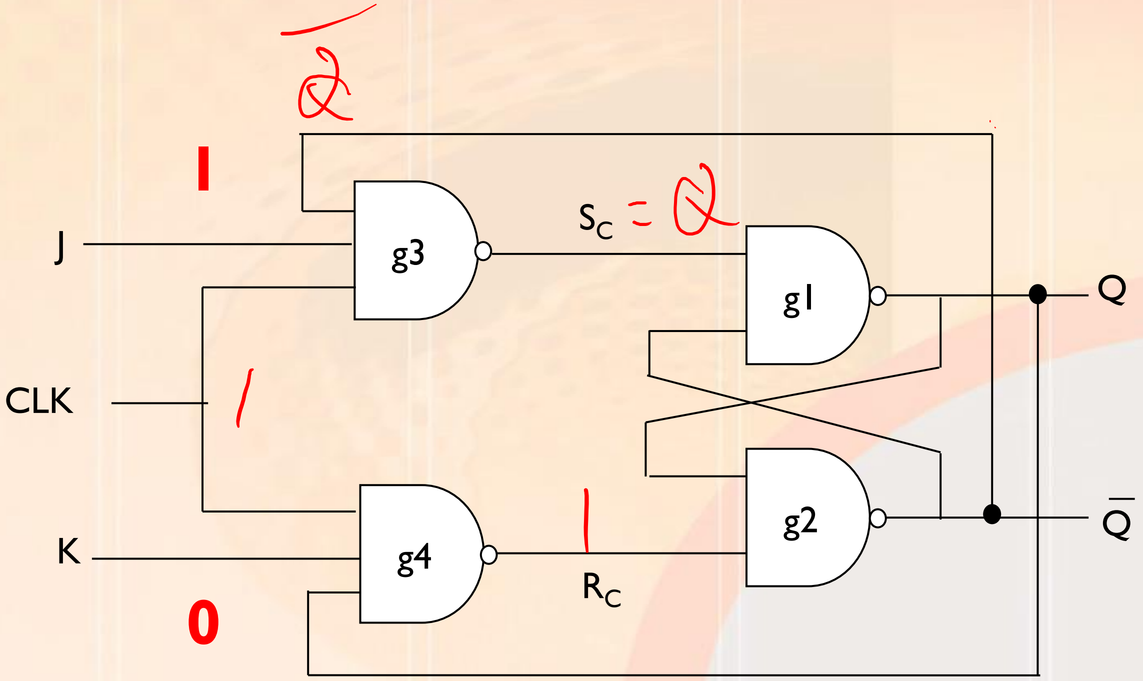
$Q(\tau) = \boxed{0}$

$\bar{Q}(\tau) = \overline{Q\bar{Q}} = \boxed{1}$

$\overline{Q\bar{Q}} = \bar{0} = 1$

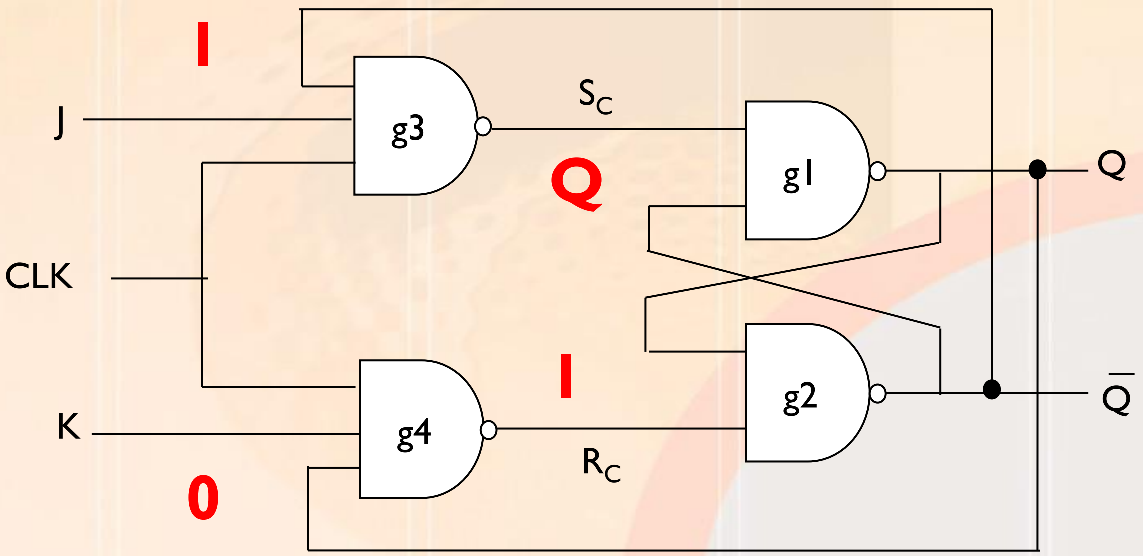
JK Flipflop

- If **J = 1** and **K = 0** …



JK Flipflop

- If **J = 1** and **K = 0** …
- Then when CLK $\rightarrow$ 1, R_C = 1, $S_C \rightarrow \overline{\overline{Q}} = Q$
- Since $S_C = Q$, $Q(\tau) \rightarrow \overline{Q\,\overline{Q}} = 1$ and since $R_C = 1$, $\overline{Q}(\tau) \rightarrow 0$
- Hence $Q(\tau) \rightarrow 1$ and $\overline{Q}(\tau) \rightarrow 0$ (same as the SR flipflop)



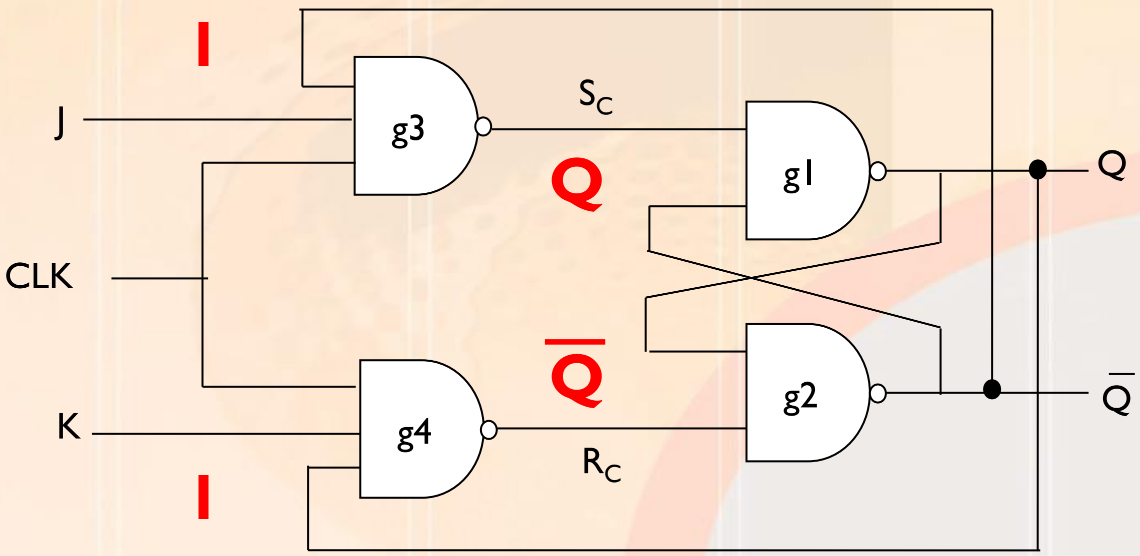
JK Flipflop

- If **J = 1** and **K = 1** …
- Then when CLK $\rightarrow$ 1, $S_C \rightarrow Q$ and $R_C \rightarrow \overline{Q}$
- Hence $Q(\tau) \rightarrow \overline{Q\,\overline{Q(\tau)}}$ and $\overline{Q(\tau)} \rightarrow \overline{\overline{Q}\,Q(\tau)} = \overline{\overline{Q}\,(\overline{Q\,\overline{Q(\tau)}})}$

$$Q + Q\,\overline{Q(\tau)}$$

$$= Q\,(1 + \overline{Q(\tau)})$$

$$= Q$$



JK Flipflop

- Overall, we can state that the operation of the JK flipflop is the same as that of the SR flipflop except for J = K = 1.

- In the case of the SR flipflop this resulted in an indeterminate state.

- In the case of the JK flipflop, this condition acts as a toggle flipflop. In other words, it complements the output.

JK Flipflop

- Hence, we can derive the following **state table for the synchronous JK flipflop:**

J	K	Q	Q(τ)	
0	0	0	**0**	Q
0	0	1	**1**	
0	1	0	**0**	0
0	1	1	**0**	
1	0	0	**1**	1
1	0	1	**1**	
1	1	0	**1**	$\overline{Q}$
1	1	1	**0**	

JK Flipflop

- The **concise form of the state table** is thus:

J	K	Q	Q(τ)
0	0	0	**0**
0	0	1	**1**
0	1	0	**0**
0	1	1	**0**
1	0	0	**1**
1	0	1	**1**
1	1	0	**1**
1	1	1	**0**

J	K	Q(τ)
0	0	**Q**
0	1	**0**
1	0	**1**
1	1	$\overline{Q}$

JK Flipflop

- By carefully analysing the state table, we can derive the following **requirements table** for the JK flipflop, where X represents don't care terms:

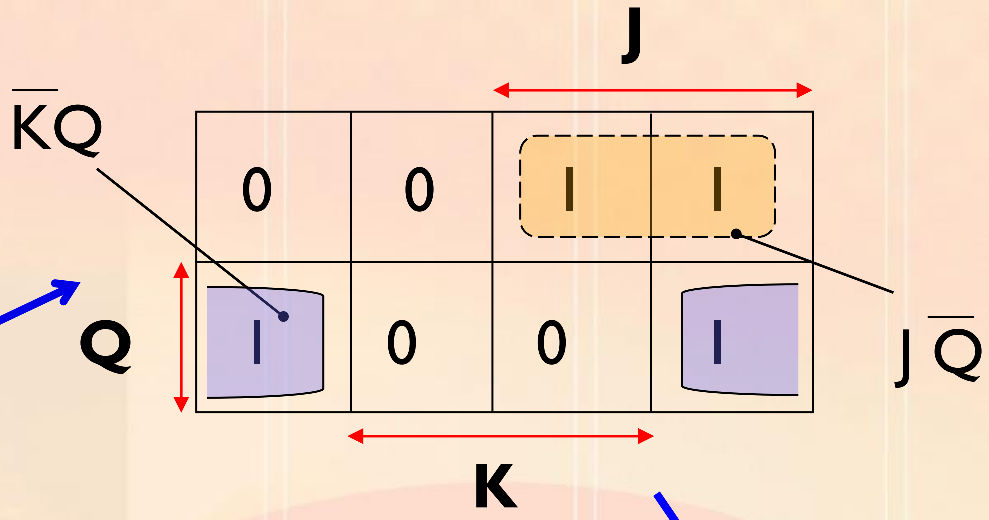
J	K	Q	Q(τ)
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

Operation	J	K
Stay at 0	0	X
Stay at 1	X	0
Go to 0	X	1
Go to 1	1	X

JK Flipflop

- The **next state equation** for the JK flipflop can be derived as follows:

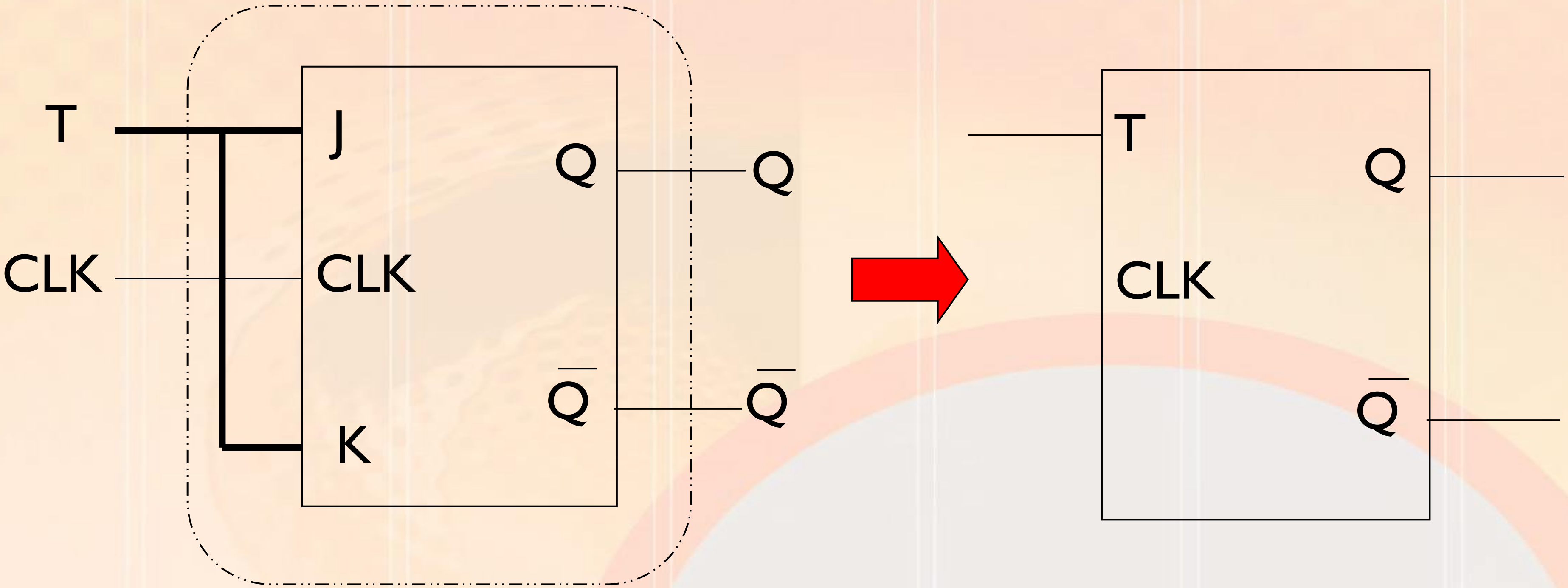
J	K	Q	Q(τ)
0	0	0	**0**
0	0	1	**1**
0	1	0	**0**
0	1	1	**0**
1	0	0	**1**
1	0	1	**1**
1	1	0	**1**
1	1	1	**0**



$$Q(\tau) = J\overline{Q} + \overline{K}Q$$

T (Toggle) Flipflop

- The toggle flipflop is a modified version of the JK flipflop, where both inputs are simply tied together as follows:



T (Toggle) Flipflop

- Note that by connecting J and K to T, we are restricting the J and K inputs to be either both 0 or 1. The condition $J \neq K$ can never occur.

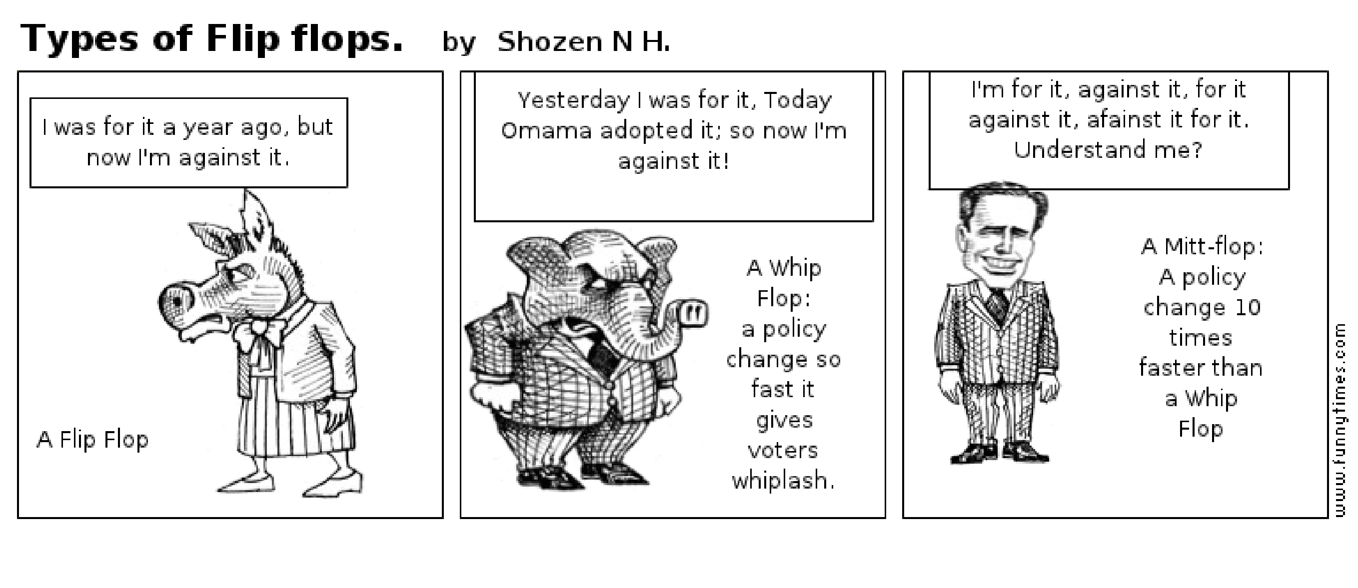
- Hence the following **state table**:

T	J	K	Q	$Q(\tau)$
0	0	0	0	**0**
0	0	0	1	**1**
1	1	1	0	**1**
1	1	1	1	**0**

T	$Q(\tau)$
0	**Q**
1	$\overline{\mathbf{Q}}$

T (Toggle) Flipflop

- In brief, the T flipflop complements the output when T = 1, otherwise the output remains the same when T = 0.



T (Toggle) Flipflop

- The requirements table for the T flipflop is:

Operation	T
Stay at 0	**0**
Stay at 1	**0**
Go to 0	**1**
Go to 1	**1**

Flipflops v Latches

- Latches are very similar devices to flipflops – they are constructed in the same fashion and have the same basic operating principles. However there are two notable differences.

- Firstly, and a relatively minor different, latches tend to use an Enable (or Control) input instead of the Clock input associated with the flipflops.

- Both work using the same principle, whereby the output of the device can only change depending on the Enable input or the Clock input being set HIGH (typically).

- However, the Clock tends to be a regular periodic signal, while the Enable key does not have to be.

Flipflops v Latches

- The second and fundamental difference relates to when the actual output is allowed to change.

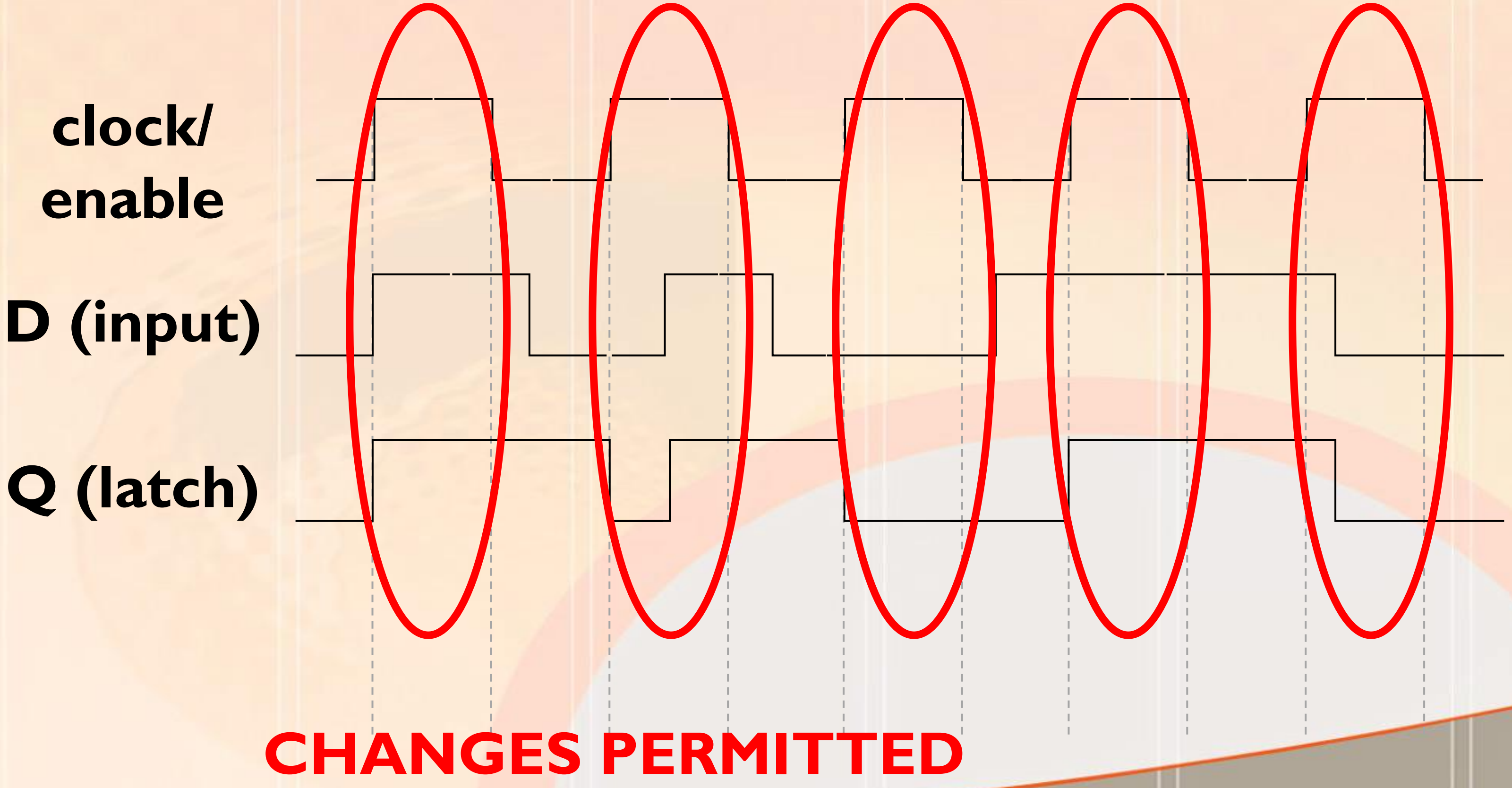
- In the case of the **latch** the **output can change at any instant in time while the Enable input is set HIGH**.

- In **flipflops**, the **output can only change when the Clock is in transition between two states**, either LOW to HIGH or HIGH to LOW.

- This difference has a significant impact on the timing of both devices.

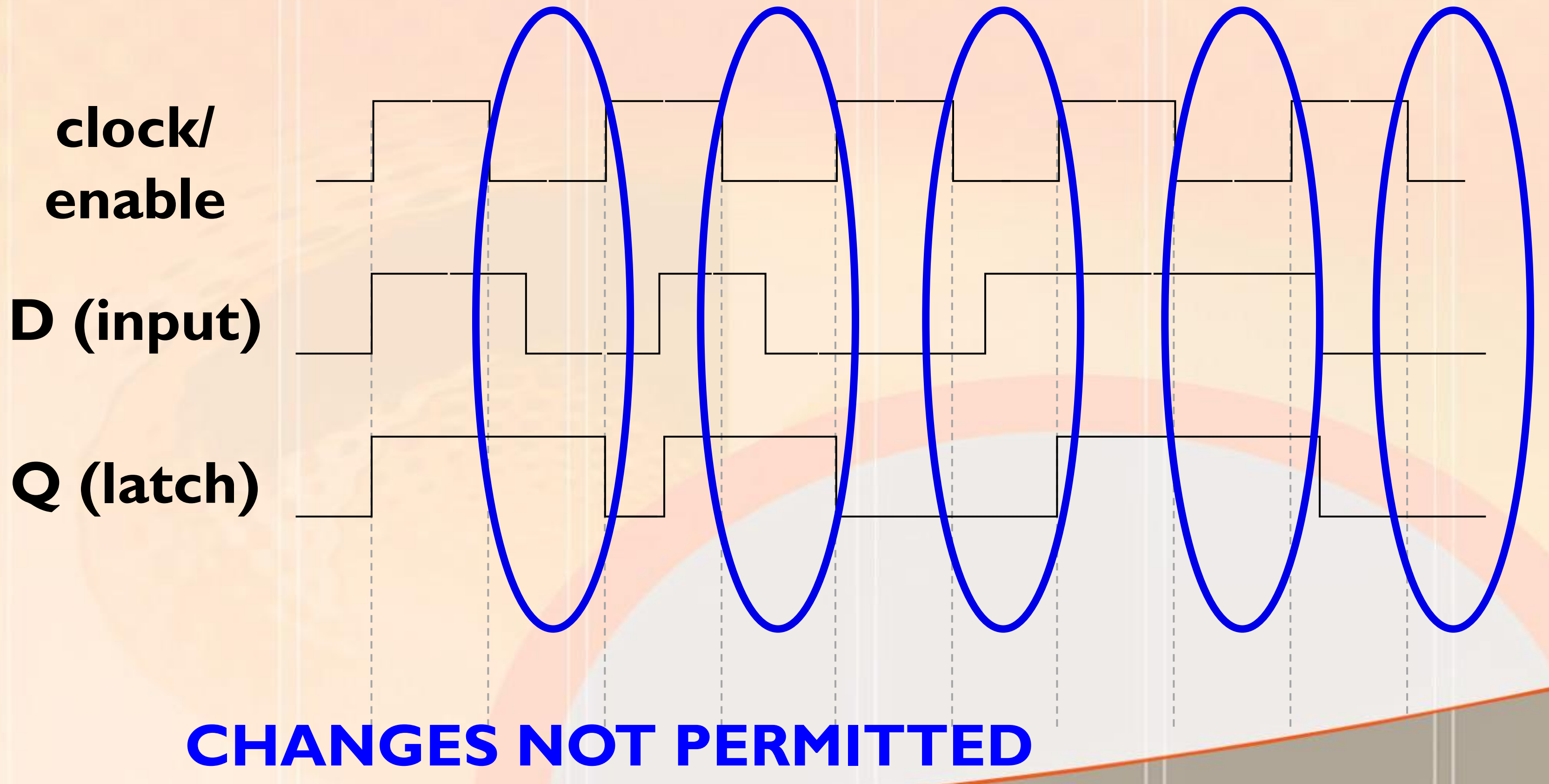
Flipflops v Latches

- By way of example, consider the timing associated with a D flipflop, using a positive-edge clock transition (i.e. from 0 to 1), and a D latch:



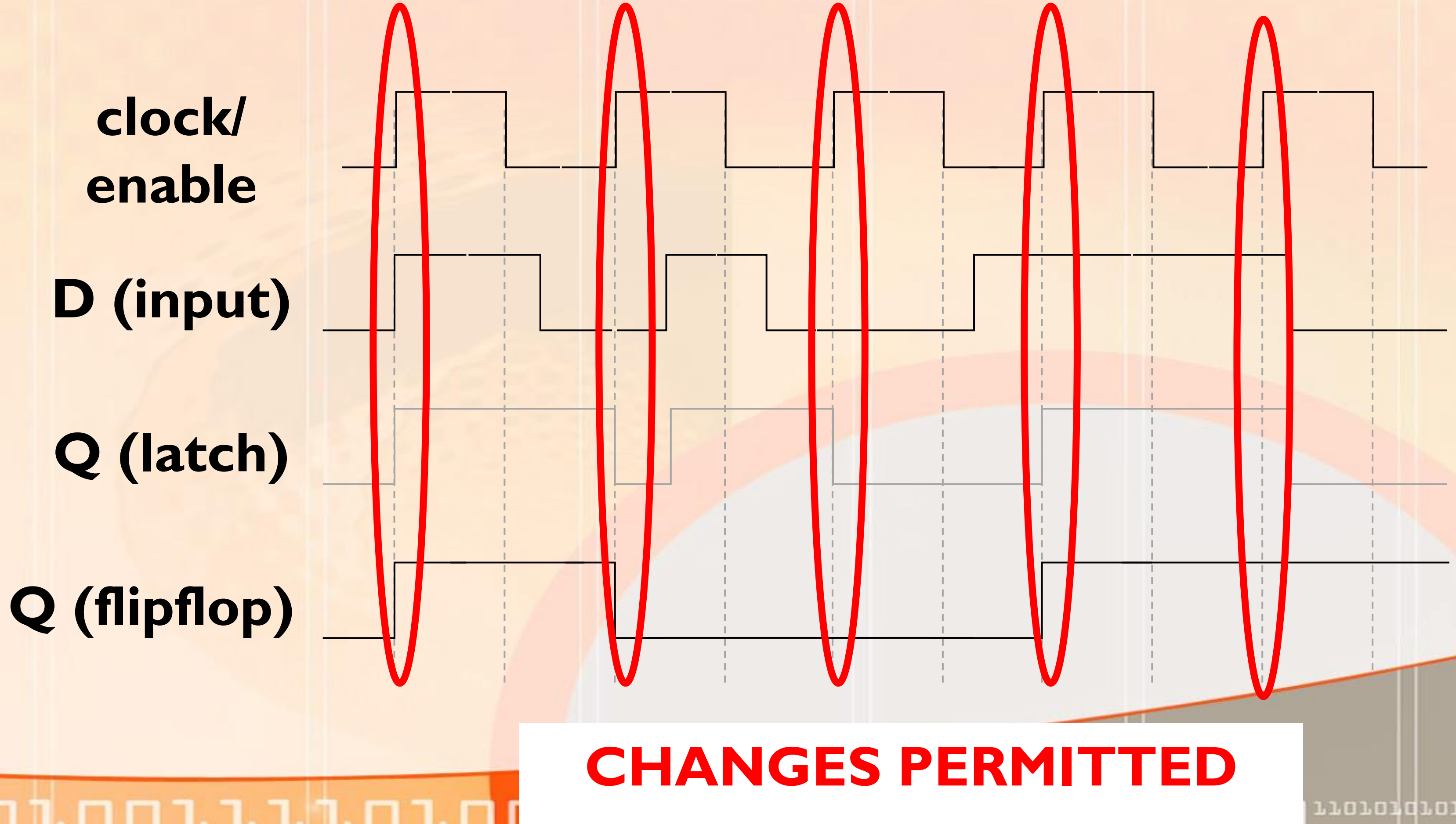
Flipflops v Latches

- By way of example, consider the timing associated with a D flipflop, using a positive-edge clock transition (i.e. from 0 to 1), and a D latch:



Flipflops v Latches

- By way of example, consider the timing associated with a D flipflop, using a positive-edge clock transition (i.e. from 0 to 1), and a D latch:



Flipflops v Latches

- The symbol for an edge-triggered flipflop (for example the D flipflop) looks like:

Flipflops v Latches

- The symbol for an edge-triggered flipflop (for example the D flipflop) looks like: