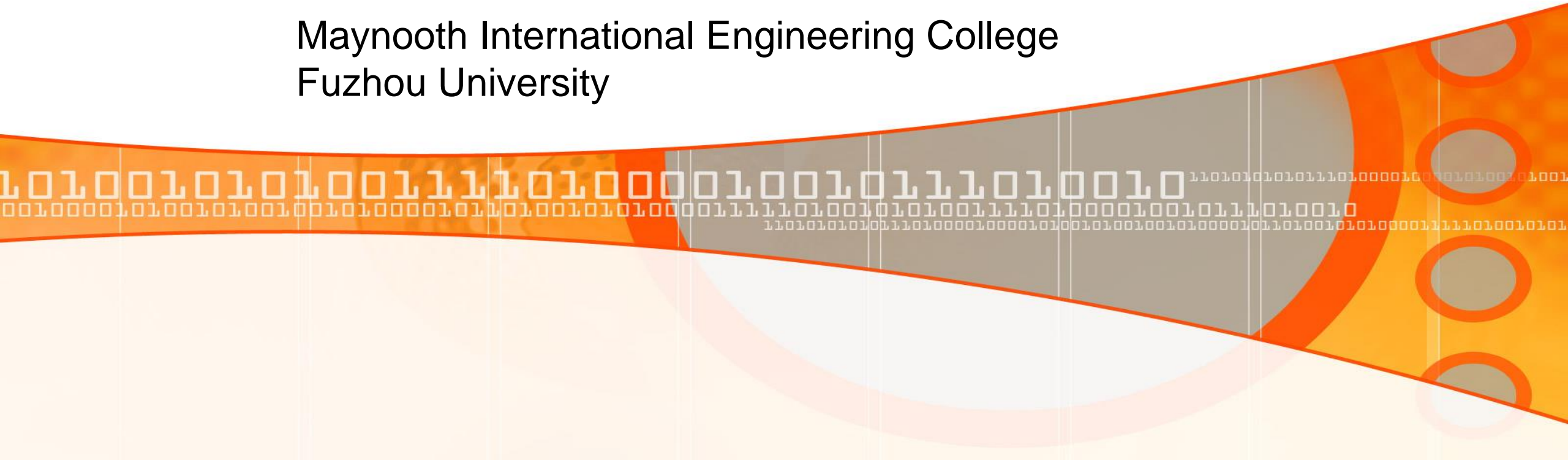


EE103 Digital Systems 1

Wong Chin Hong

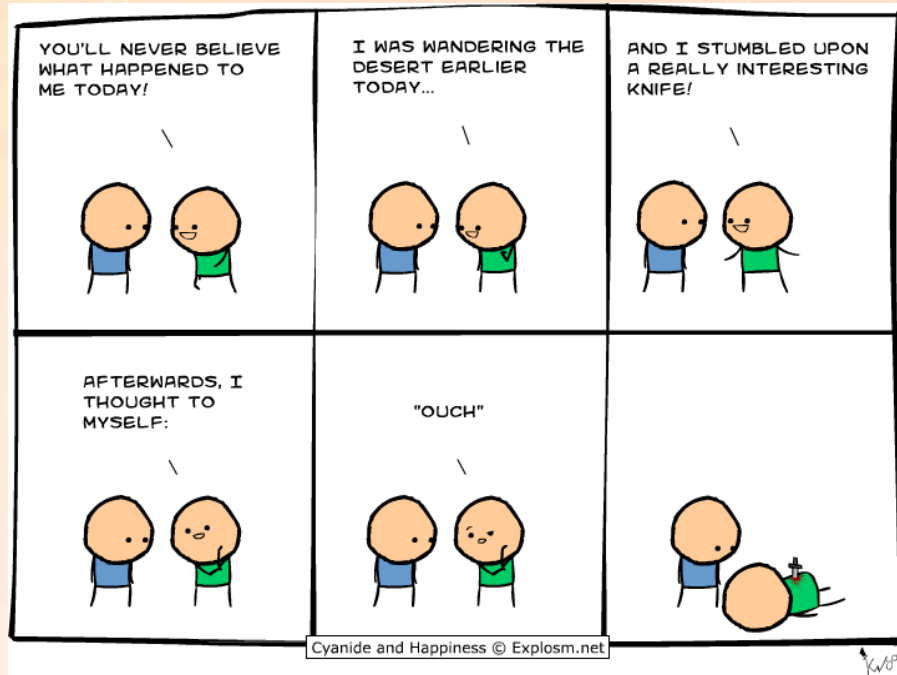
106

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 looked in detail at the SR, D, JK and T flipflops ...



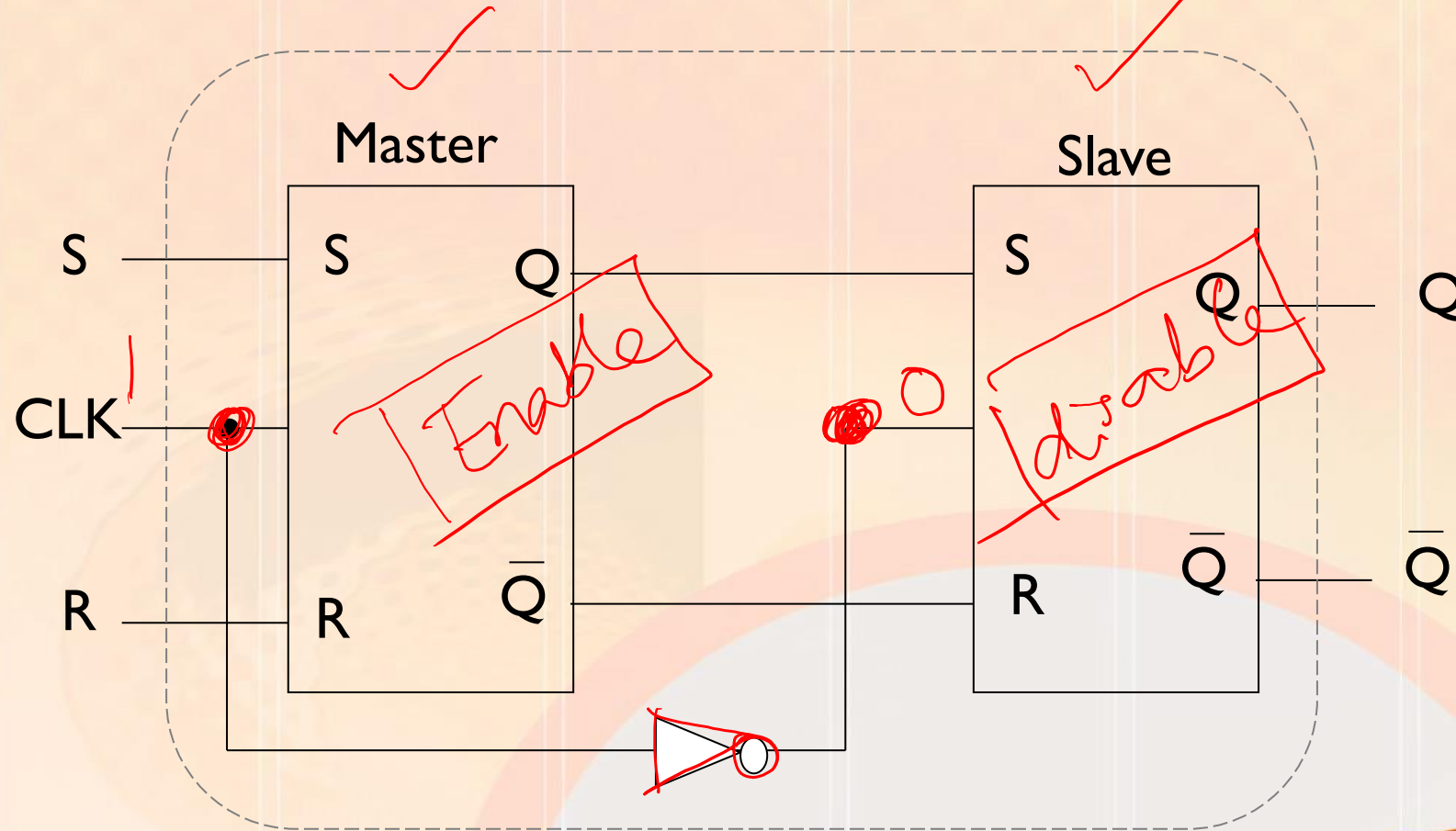
TODAY, we are going to look at Master-Slave flipflops, asynchronous inputs and introduce counters ...

Master-Slave Flipflops

- Thus, it is possible for the latch output to change several times during this period.
- As already indicated, an edge-triggered flip-flop can resolve this issue.
- Alternatively, we could use a two-stage latch which ensures that the output does not change state until after the *trailing edge* of the clock pulse.
- This configuration is known as a master-slave flipflop.

Master-Slave Flipflops

- By way of illustration, consider the **Master-Slave SR flipflop**:

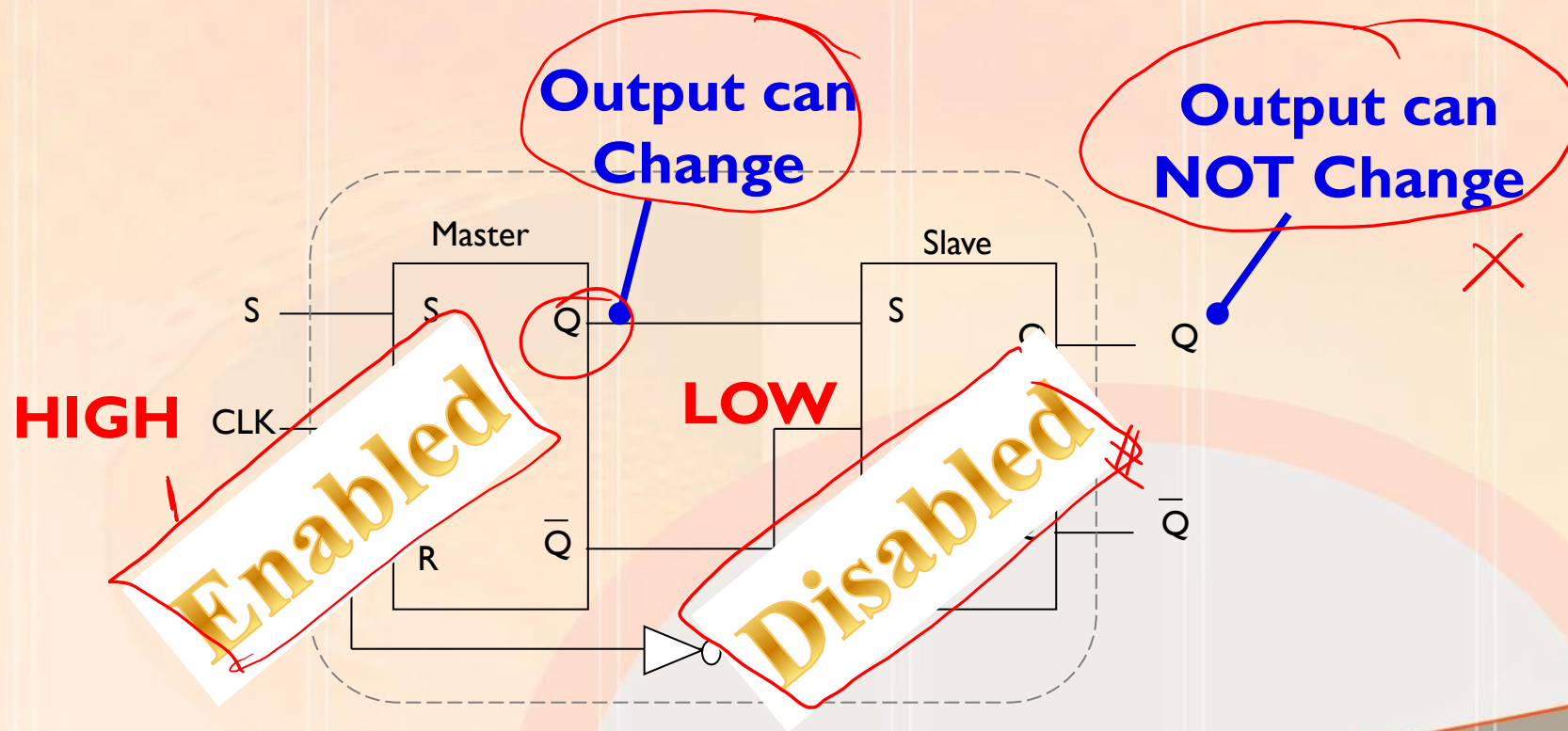


Master-Slave Flipflops

- One circuit acts as the master and the other as the slave, hence the name!
- The operation of this device is relatively straightforward.

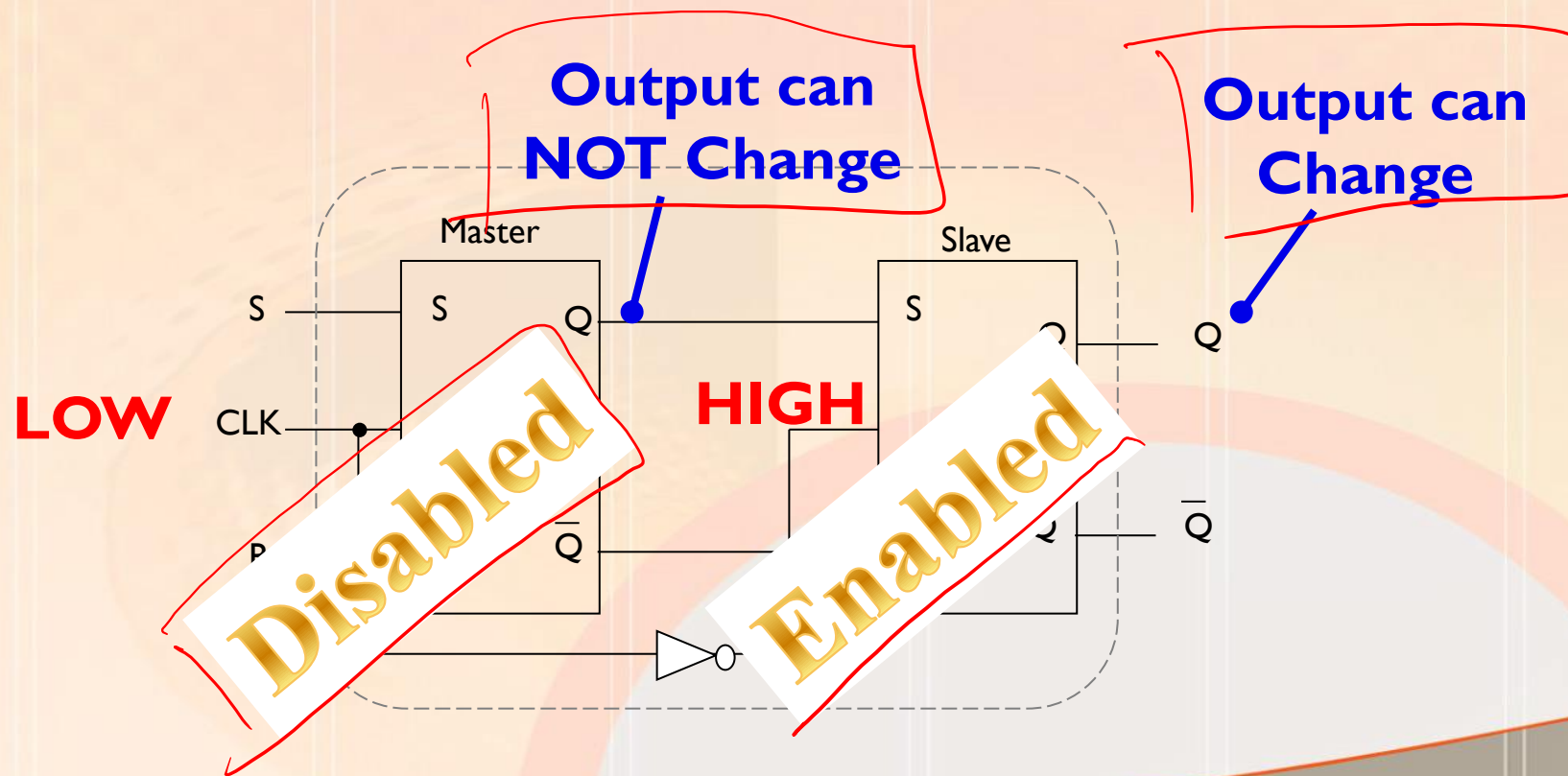
Master-Slave Flipflops

- When clock is HIGH, the inputs to the Master flipflop can affect its outputs. The Slave device is disabled and, as such, its outputs remain unchanged.



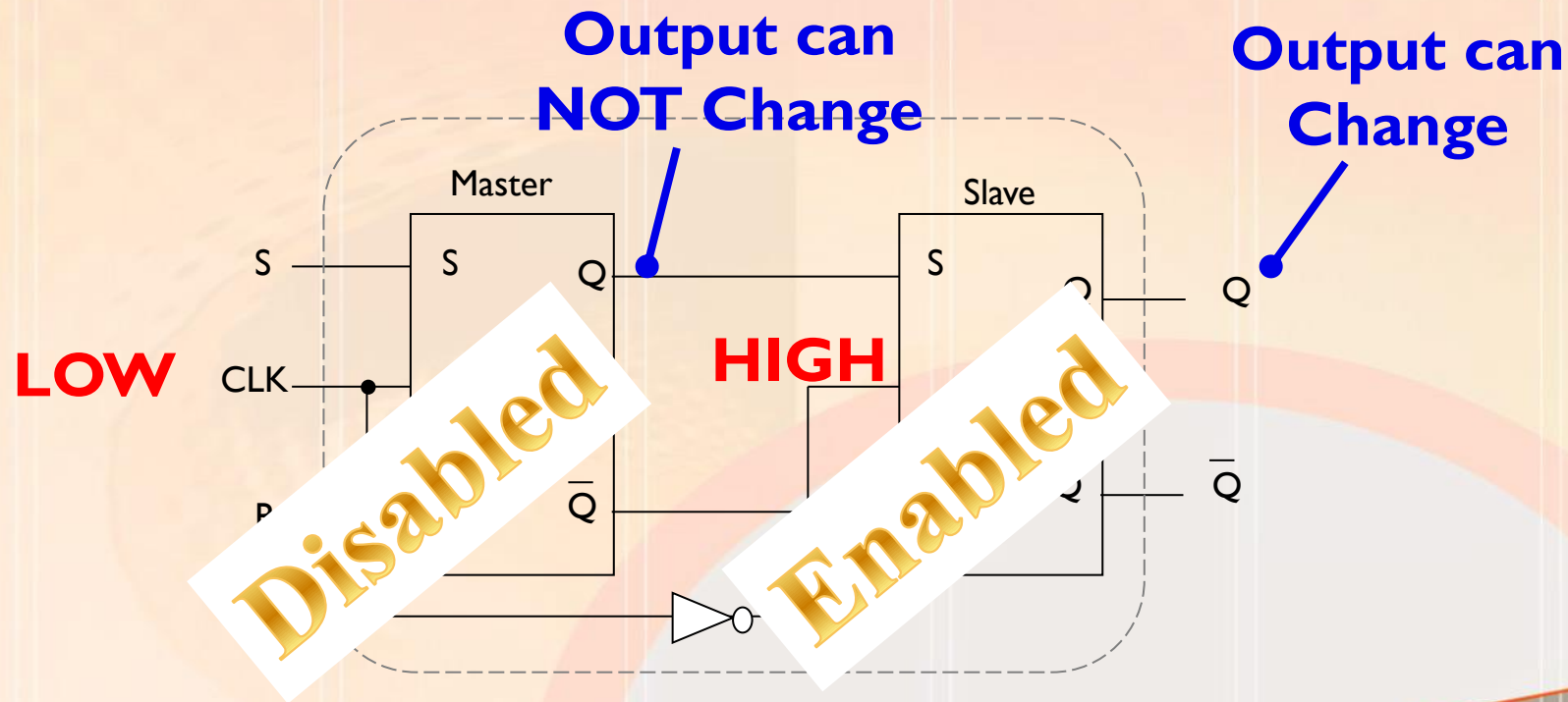
Master-Slave Flipflops

- When clock goes LOW, the Slave is now enabled and the outputs of the Master flipflop and, hence, the inputs to the Slave flipflop, now affect the outputs of the latter.



Master-Slave Flipflops

- The Master flipflop is now disabled so its outputs are unchanged during this LOW clock cycle.



Master-Slave Flipflops

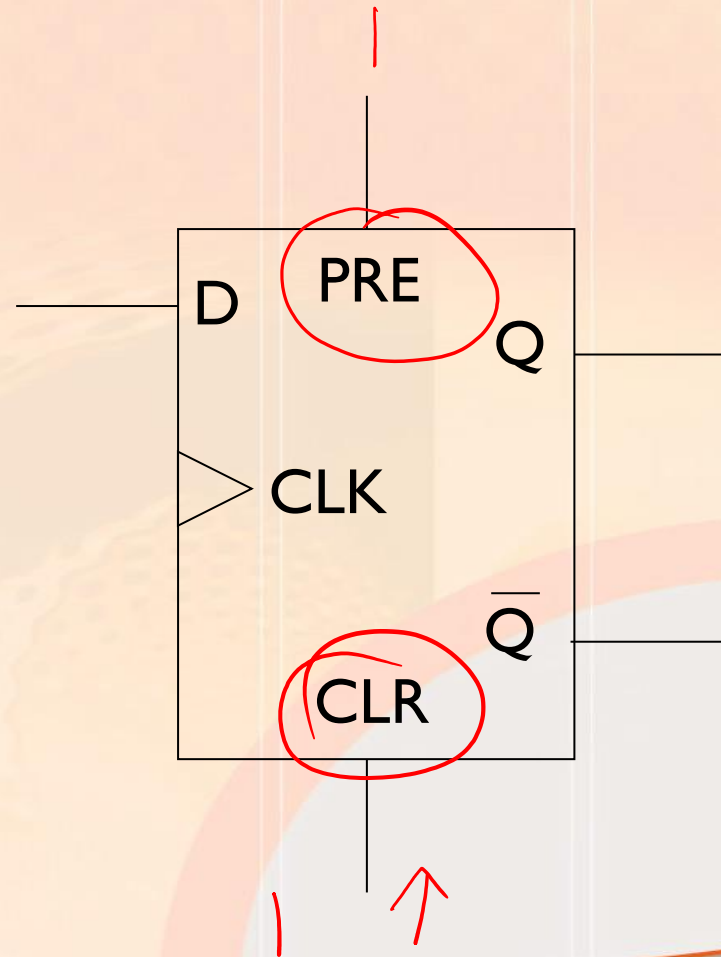
- In this structure, the **outputs** of the overall master-slave device **can only change once in one full clock period**.

Asynchronous Inputs

- Flipflops often have additional inputs that allow them to be set and reset **independently of the clock input**.
- Such inputs are known as **asynchronous inputs**.
- The inputs that asynchronously set the flipflop are called **preset.** = 1
- The inputs that asynchronously reset the flipflop are called **clear.** = 0
- When power is turned on to a digital system, the flipflop states are randomly set. The asynchronous inputs are used to correctly initialise the digital system before the normal clocked operation begins.

Asynchronous Inputs

- The symbol for a D flipflop (for example) with asynchronous inputs looks like:



Application Examples

- There are various examples of applications that use flipflops.
- In the next sections of the notes, we will look in detail at two such applications, namely **Counters** and **Registers** (data storage).
- Two other common applications are now outlined ...

Application Examples

Contact Bounce ...

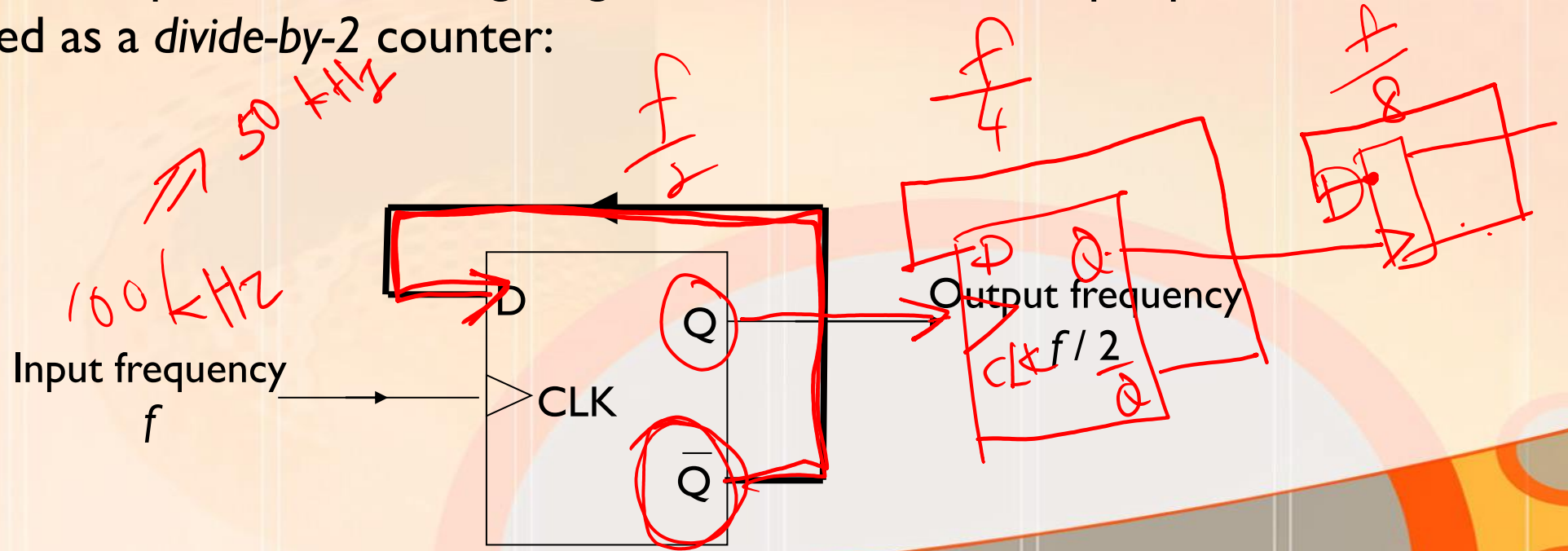
- Flipflops can also be used to eliminate the effect of contact bounce.
- When a switch is flicked, it physically vibrates and/or bounces for a short time before the contact is solid.
- This bouncing lasts a short time but can have undesirable effects in a digital system.
- By connecting the switch to a flipflop, the effect of the bounce is removed and a clean switching signal is achieved.



Application Examples

Frequency Division ...

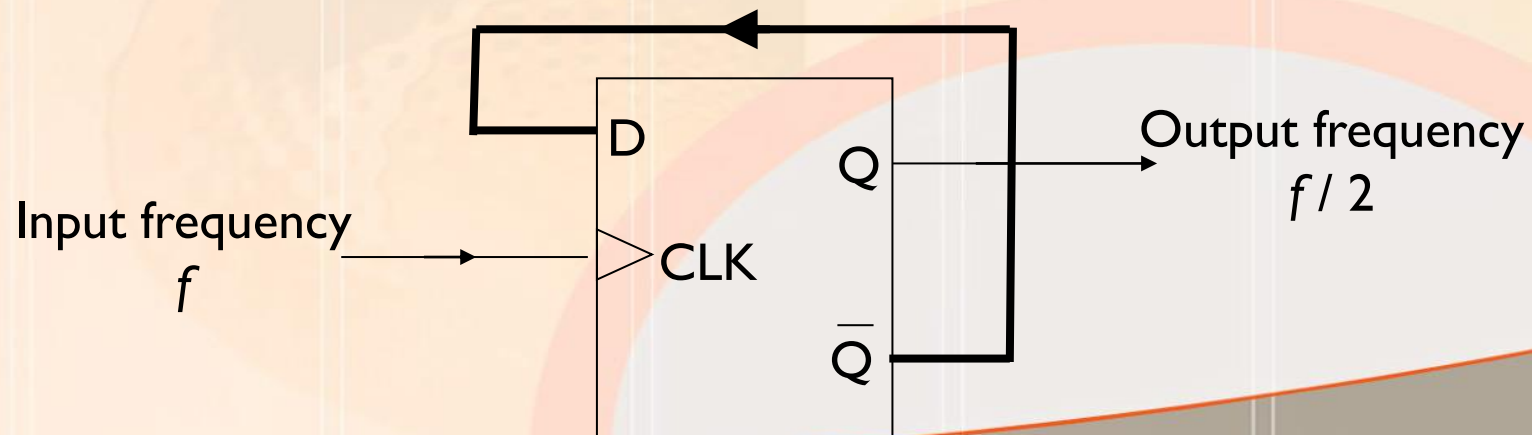
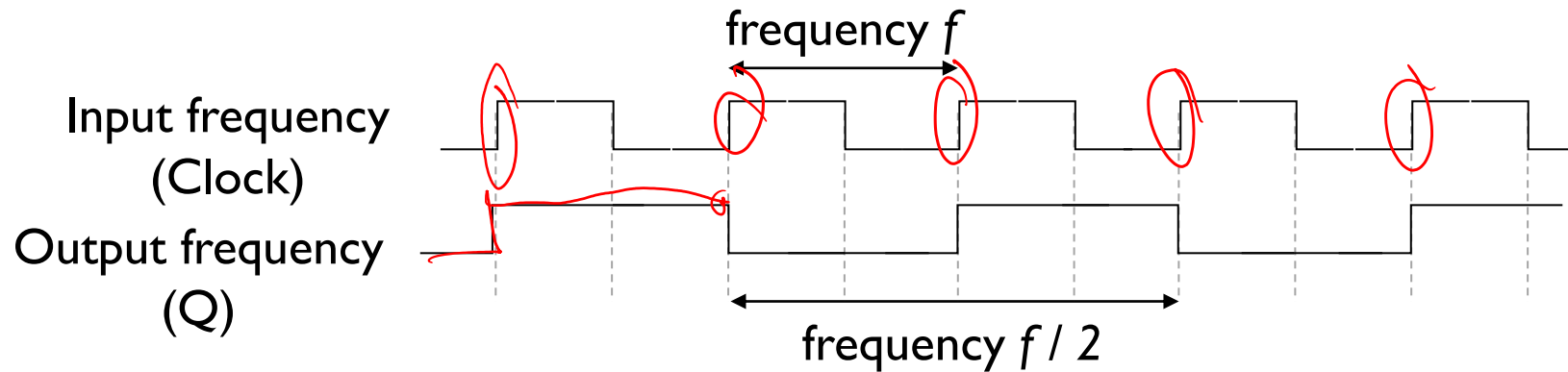
- Flipflops can also be used to divide a clock frequency by 2^n for any integer n .
- For example, the following diagram shows how a D-flipflop can be used as a *divide-by-2* counter:



Application Examples

Frequency Division ...

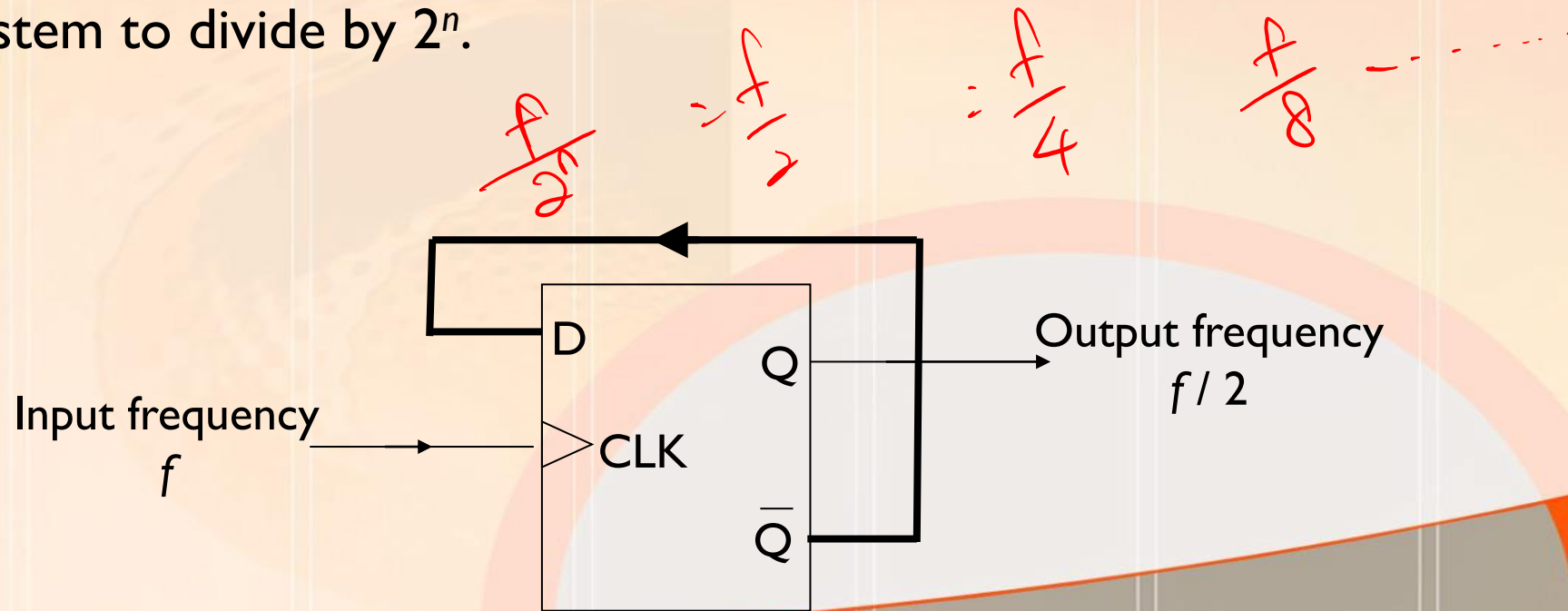
- The input (clock) and output waveforms for this counter are:

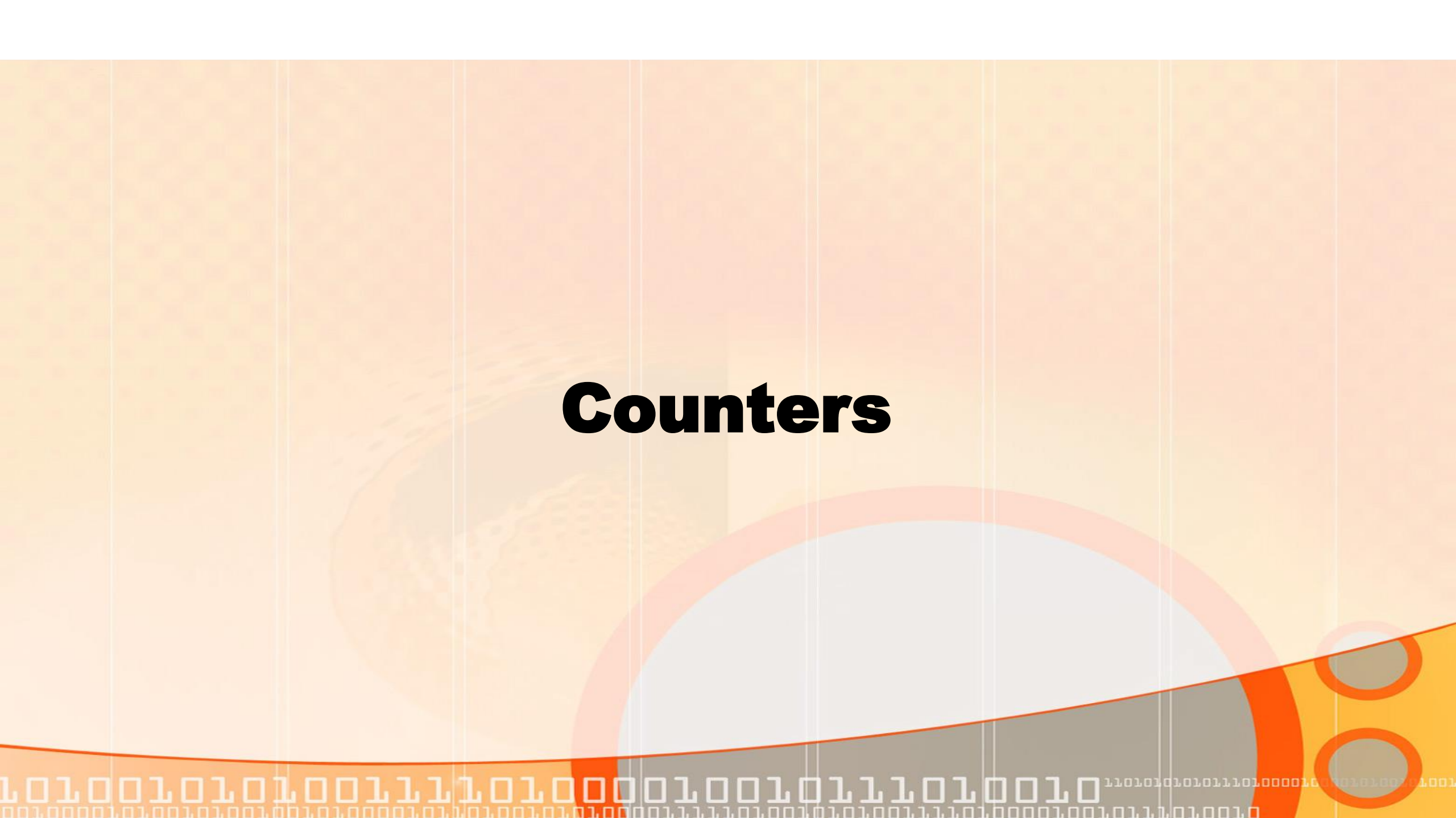


Application Examples

Frequency Division ...

- Clearly, we can see how the input frequency signal has been divided by 2.
- If we have a sequence of such devices, we can then easily build a system to divide by 2^n .



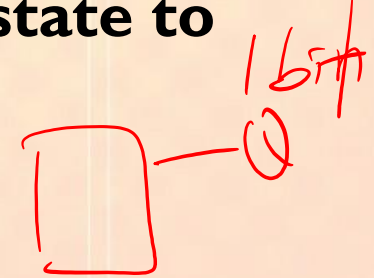


Counters

Counters



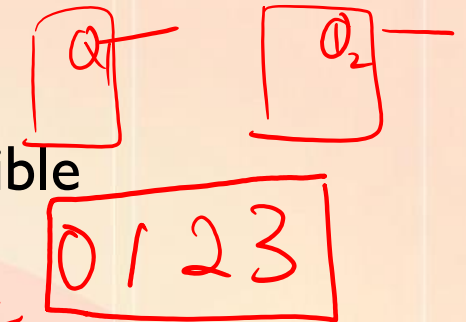
- A counter is an array of flipflops that advances from state to state in response to an event.



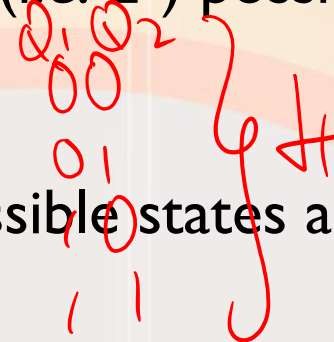
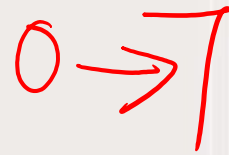
- This event is usually a single cycle of a clock waveform.

- Each flipflop represents a single bit.

- Thus a counter with two flipflops can have 4 (i.e. 2^2) possible outputs or states.



- With three flipflops, it can have 8 (i.e. 2^3) possible states and so on.



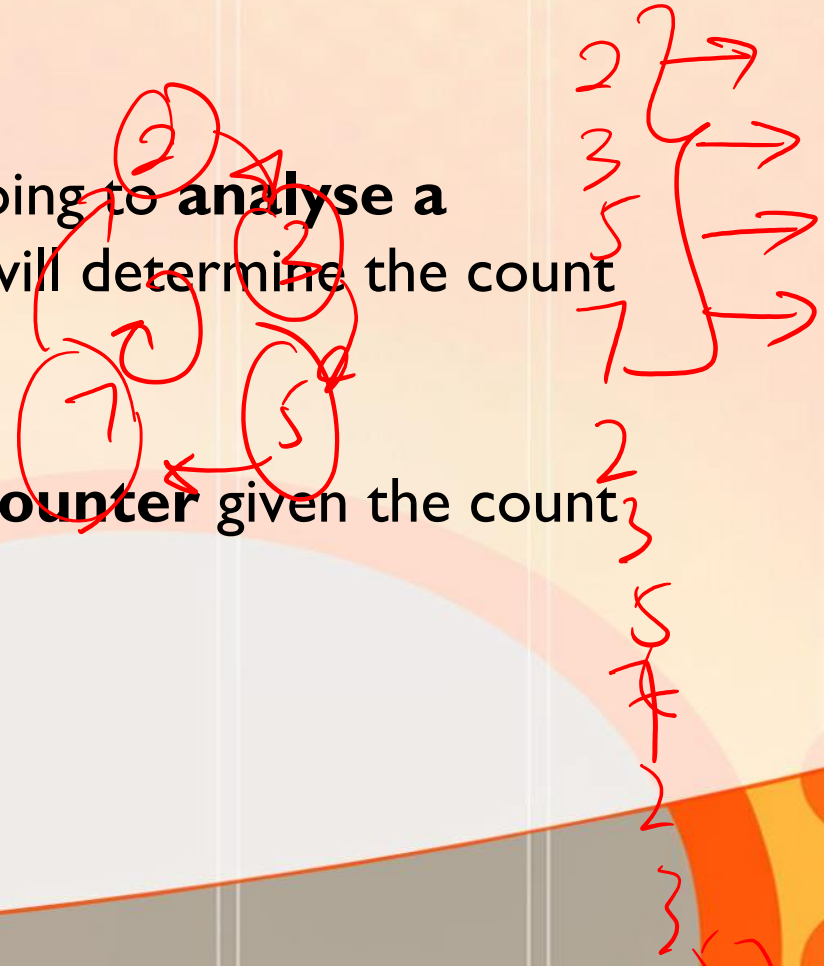
Counters

- Counters have numerous applications in digital systems:
- For example, they could be used to build the circuit for a digital watch or a digital alarm clock.
- They could also be used as a driver for a set of traffic lights, which is based on a repetitive fixed sequence of changes (or states).
- They could also be used as a program counter that steps through the addresses (or instructions) of a software program.



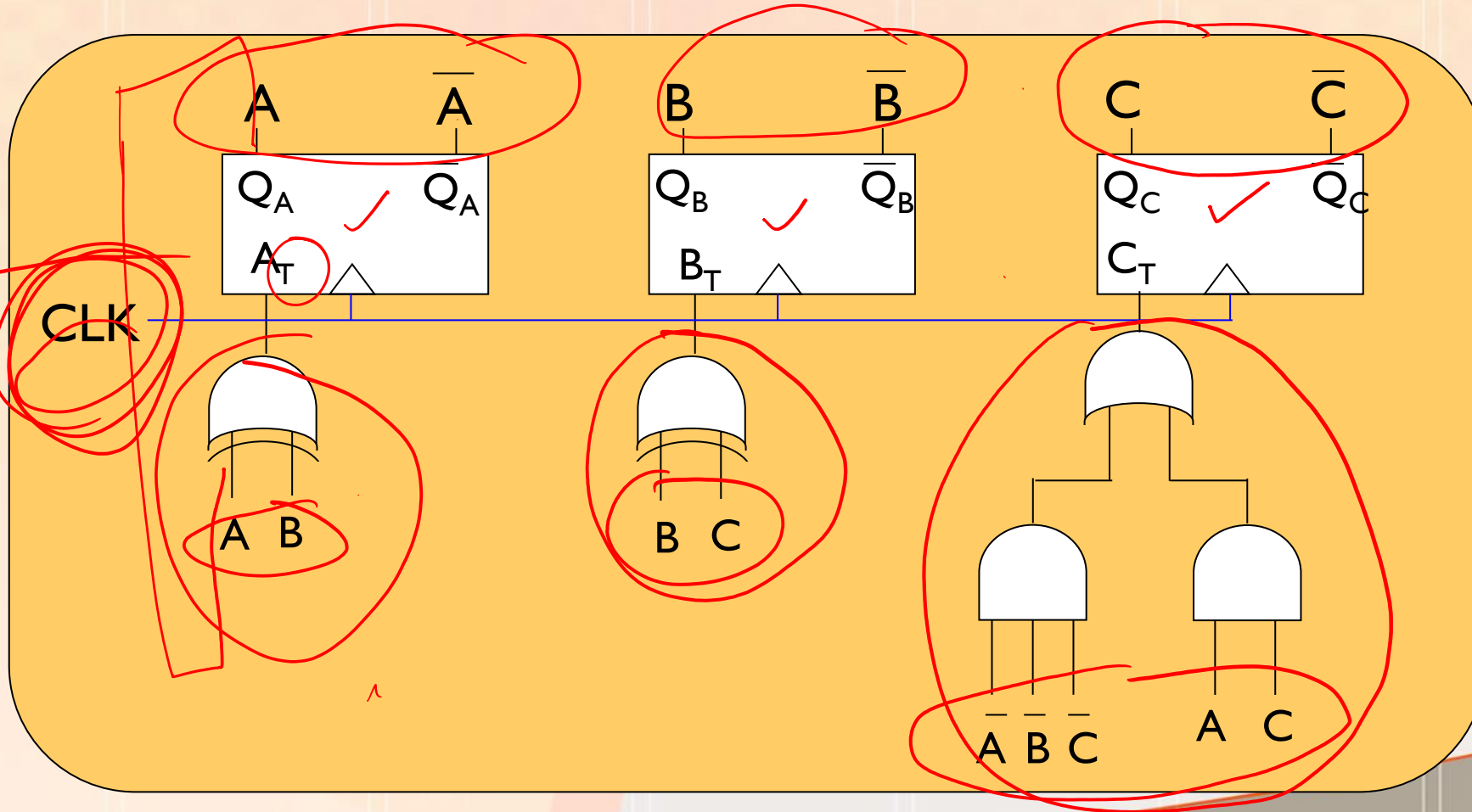
Counters

- The number of states through which a counter cycles before returning to a starting state (not necessarily zero) is called the **Modulo** (or **MOD**) of the counter.
- In the next section of the notes we are going to **analyse a counter** to see how it operates, i.e. we will determine the count sequence for a given design.
- Then we will examine how to **design a counter** given the count sequence.



Analysing a Counter

- Ex. 7.1 Determine the operation of the following circuit:



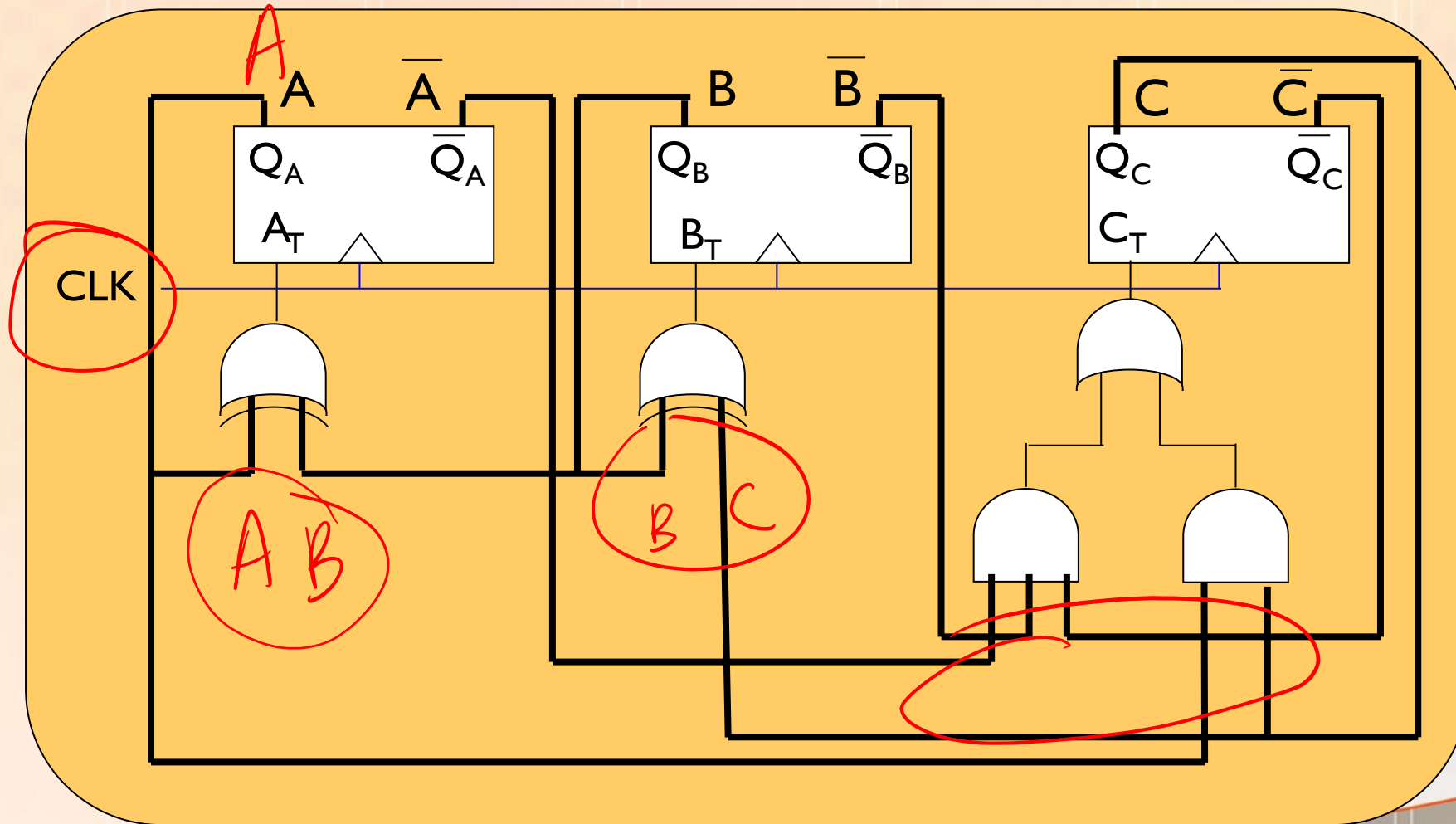
3 bits.
0
↓
8 possible states
7



Analysing a Counter

- Note that the parameters A, B and C are in fact the output states of the flipflops.
- This system has technically **only one input**, i.e. the clock signal.
- The current representation of the circuit is convenient for analytical purposes.
- In reality the circuit should be presented as follows:

Analysing a Counter



Analysing a Counter

- In order to fully analyse the circuit, we need to consider all combinations of states (or outputs) of the flipflops, i.e. from $ABC = 000$ to 111 .
- From this we can determine the count sequence for the circuit.
- In other words, we can determine the sequence of states that the counter will transition through before repeating.

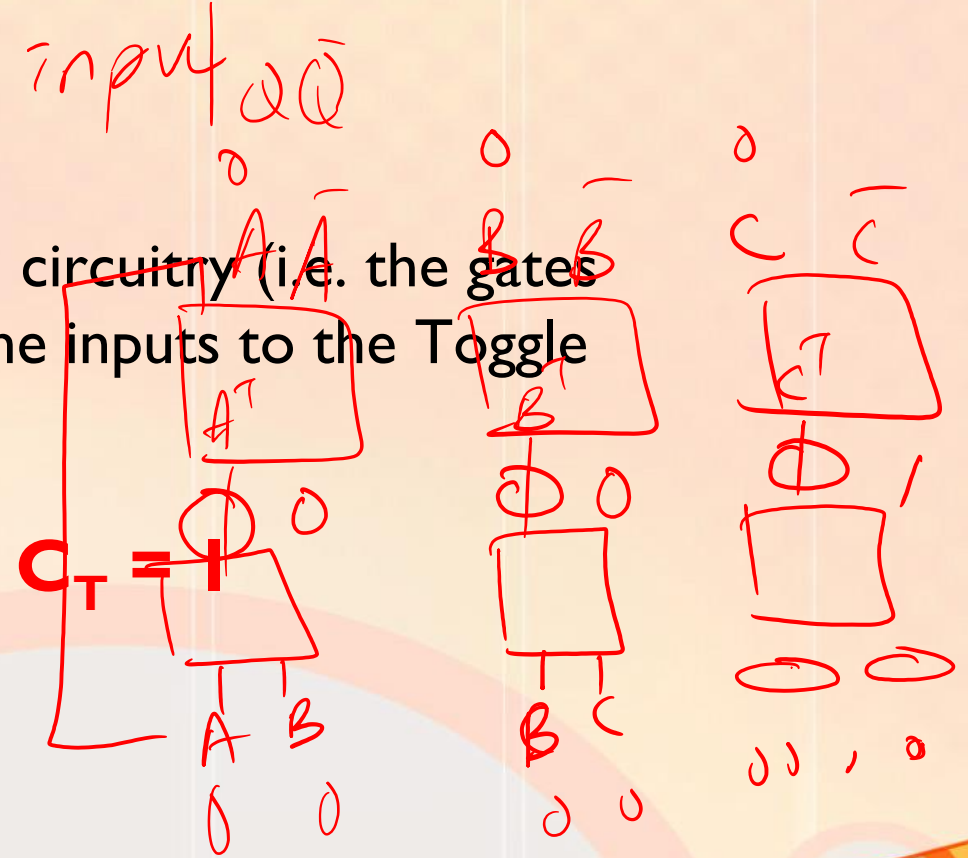
Analysing a Counter

- So, taking the first state combination:

$$\mathbf{ABC = 000}$$

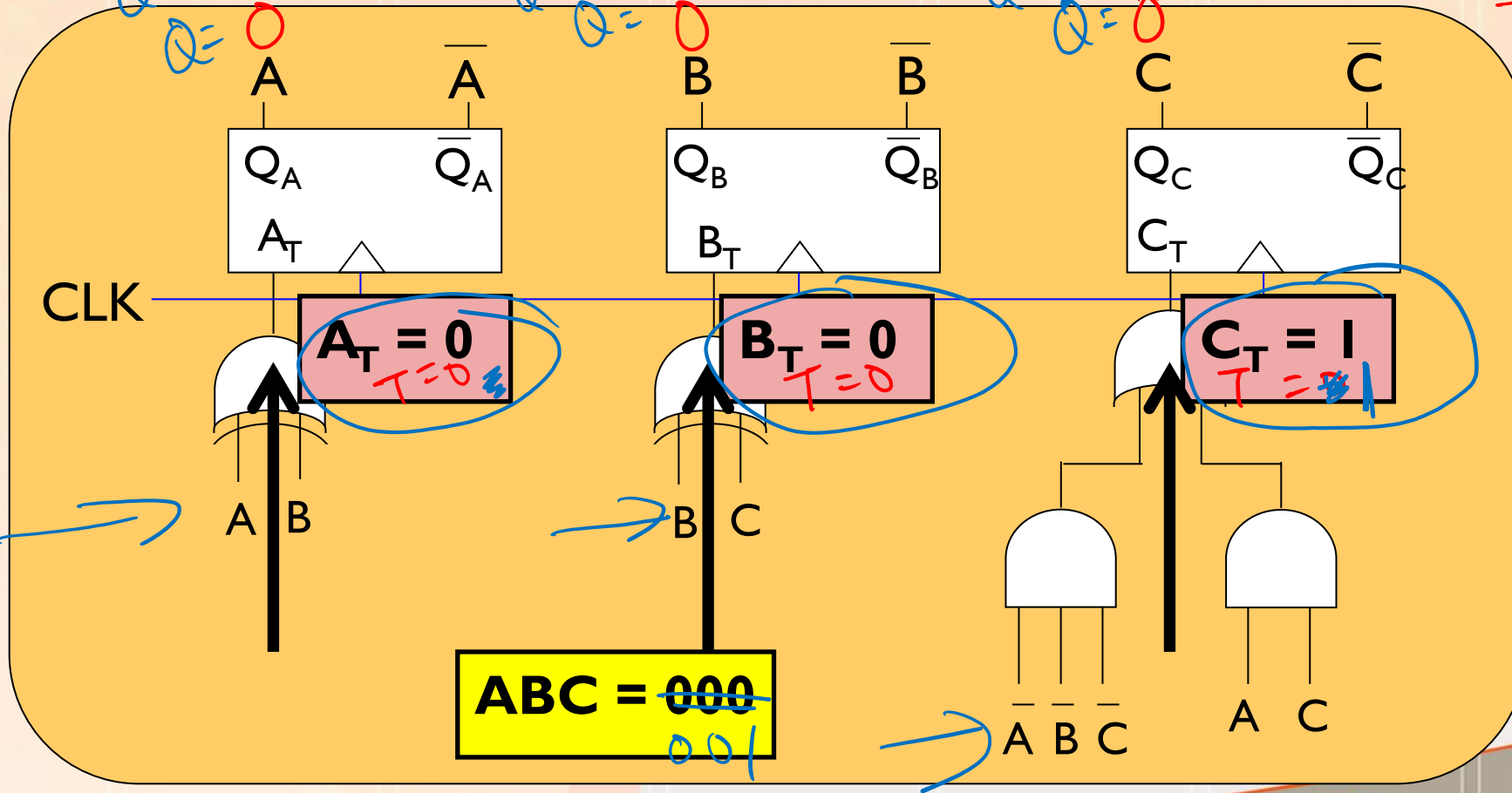
- We can determine from the combinational circuitry (i.e. the gates leading to the inputs of the flipflops) that the inputs to the Toggle flipflops will be:

$$\mathbf{A_T = 0, B_T = 0 \text{ and } C_T = 1}$$



Analysing a Counter

- Ex. 7.1 Determine the operation of the following circuit:



T	Q _A
0	Q
1	Q



Analysing a Counter

- Recall the operation of a Toggle flipflop:

T = 0 implies **no change** while

T = 1 **complements** the existing output.

Analysing a Counter

- Hence, we can determine the next output for the circuit to be:

$$\mathbf{A(\tau) = 0}$$

(A = 0, T = 0 \Rightarrow no change)

$$\mathbf{B(\tau) = 0}$$

(B = 0, T = 0 \Rightarrow no change)

$$\mathbf{C(\tau) = 1}$$

(C = 0, T = 1 \Rightarrow complement existing output, i.e. $C \rightarrow 1$)

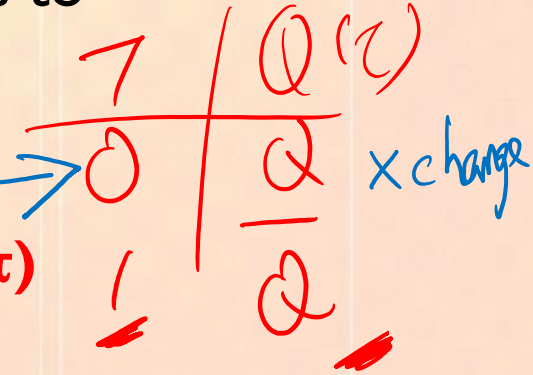
- Thus the **next output** for the circuit is:

$$\mathbf{A(\tau)B(\tau)C(\tau) = 001}$$

Analysing a Counter

- We repeat this process for the remaining state combinations to obtain the following table:

Present State			Flipflop Inputs			Next State		
A	B	C	$A \oplus B$ A_T	$B \oplus C$ B_T	$ABC + AC$ C_T	$A(\tau)$	$B(\tau)$	$C(\tau)$
0	0	0	0	0	1	0	0	1
0	0	1	0	1	0	0	1	1
0	1	0	1	1	0	1	0	0
0	1	1	1	0	0	1	1	1
1	0	0	1	0	0	0	0	0
1	0	1	1	1	1	0	1	0
1	1	0	0	1	0	1	0	0
1	1	1	0	0	1	1	1	0



Analysing a Counter

- In order to visually illustrate the behavior of the counter circuit, we use a **state diagram** to show the sequence of states.
- We can extract the sequence of states from the previous table as follows ...



Analysing a Counter

- We repeat this process for the remaining state combinations to obtain the following table:

0

	Present State			Flipflop Inputs			Next State		
	A	B	C	$A \oplus B$ A_T	$B \oplus C$ B_T	$\overline{A}BC + AC$ C_T	$A(\tau)$	$B(\tau)$	$C(\tau)$
0	0	0	0	0	0	1	0	0	1
1	0	0	1	0	1	0	0	1	1
2	0	1	0	1	1	0	1	0	0
3	0	1	1	1	0	0	1	1	1
4	1	0	0	1	0	0	0	0	0
5	1	0	1	1	1	1	0	1	0
6	1	1	0	0	1	0	1	0	0
7	1	1	1	0	0	1	1	1	0

Analysing a Counter

- We repeat this process for the remaining state combinations to obtain the following table:

0
1
2
3
4
5
6
7

	Present State			Flipflop Inputs			Next State		
	A	B	C	$A \oplus B$ A_T	$B \oplus C$ B_T	$\overline{A}\overline{B}\overline{C} + AC$ C_T	$A(\tau)$	$B(\tau)$	$C(\tau)$
0	0	0	0	0	0	1	0	0	1
1	0	0	1	0	1	0	0	1	1
2	0	1	0	1	1	0	1	0	0
3	0	1	1	1	0	0	1	1	1
4	1	0	0	1	0	0	0	0	0
5	1	0	1	1	1	1	0	1	0
6	1	1	0	0	1	0	1	0	0
7	1	1	1	0	0	1	1	1	0

0
1
2
3
4
5
6
7

Analysing a Counter

- We repeat this process for the remaining state combinations to obtain the following table:

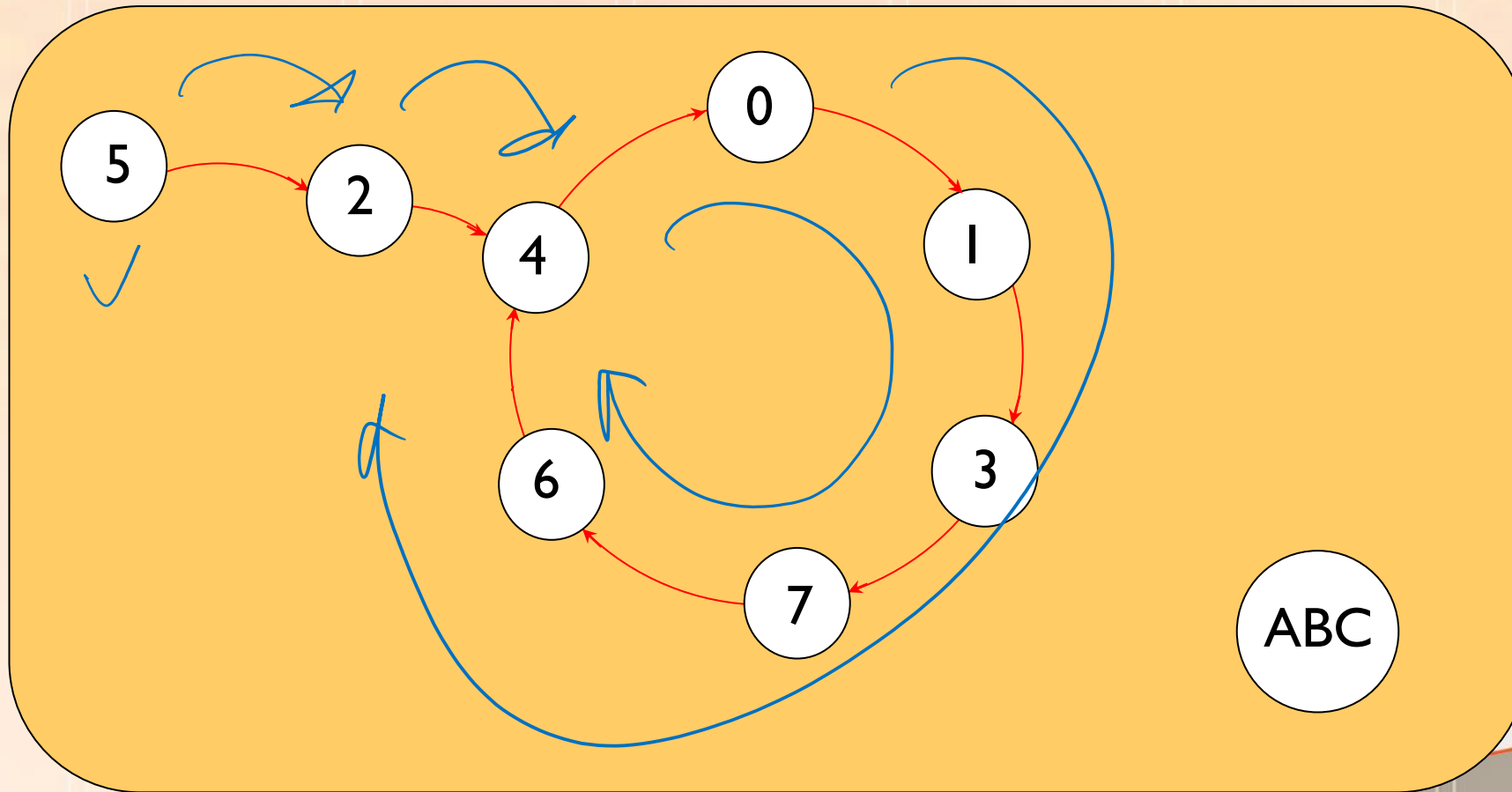
	Present State			Flipflop Inputs			Next State		
	A B C			$A \oplus B$	$B \oplus C$	$\overline{A}\overline{B}C + A\overline{B}\overline{C}$	$A(\tau)$	$B(\tau)$	$C(\tau)$
				A_T	B_T	C_T			
0	0	0	0	0	0	1	0	0	1
1	0	0	1	0	1	0	0	1	1
2	0	1	0	1	1	0	1	0	0
3	0	1	1	1	0	1	1	1	1
4	1	0	0	1	0	0	0	0	0
5	1	0	1	1	1	0	0	1	0
6	1	1	0	0	1	0	1	0	0
7	1	1	1	0	0	1	1	1	0

State 010?



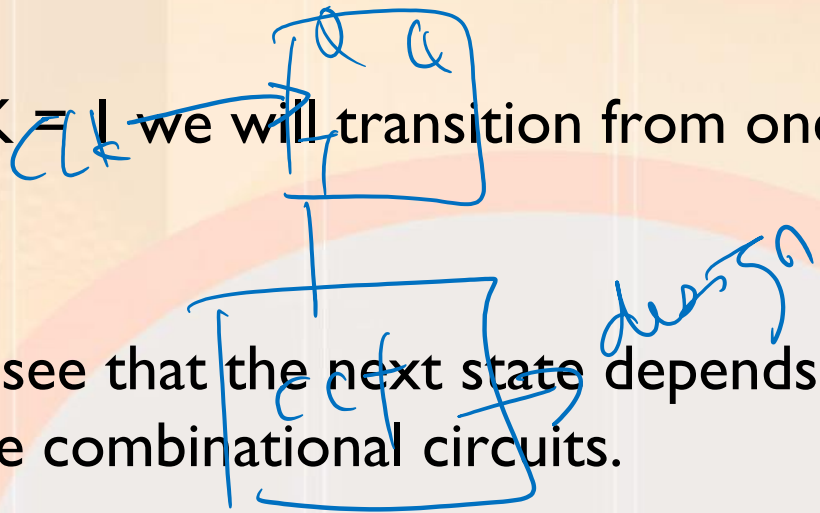
Analysing a Counter

- Hence, we can easily obtain the following state diagram:



Analysing a Counter

- Note, in the above state diagram, the **state** is given by the output combination of ABC.
- The **control variable** in this case is the clock (CLK) and is omitted from the diagram as this is the standard control variable for counters.
- In other words, when CLK = 1 we will transition from one state to the next.
- In this case we can clearly see that the next state depends primarily on the present state, unlike combinational circuits.

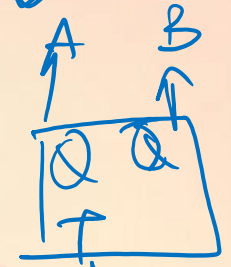


Designing a Counter

- Now, we are going to take the state sequence obtained in the previous example as our starting point and we are going to design a counter to achieve this sequence.

- Ex. 7.2 Design a MOD 6 counter with a repeating sequence 0 1 3 7 6 4, using T flipflops. Check for possible lockout.**

MOD 6 Counter



design
circuit

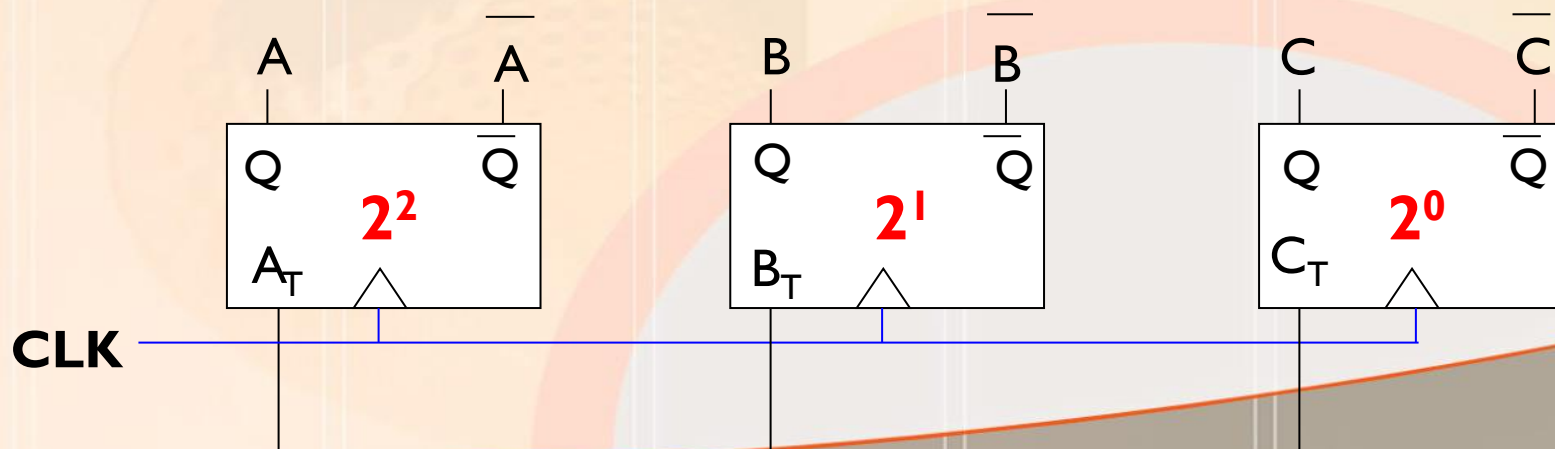
Designing a Counter

- Now, we are going to take the state sequence obtained in the previous example as our starting point and we are going to design a counter to achieve this sequence.
- **Ex. 7.2 Design a MOD 6 counter with a repeating sequence 0 1 3 7 6 4, using T flipflops. Check for possible lockout.**

*This implies that we
will count 6 states
before repeating ...*

Designing a Counter

- **STEP I** - We **need to determine the number of flipflops** required.
- Since the sequence does not contain a number greater than 7 (i.e. 111 in binary), we only need to use **3** flipflops.
- We will label these A, B and C and, arbitrarily, choose A to represent the **MSB**. Hence:



Designing a Counter

- The design is reduced to determining the logic necessary to generate A_T , B_T and C_T from the states of A, B and C to produce the new values for A, B and C, i.e.:

