COM1006 Devices and Networks (Autumn)
COM1090 Computer Architectures

Lecture #8

Sequential Circuits

Dr Dirk Sudholt
Department of Computer Science
University of Sheffield

d.sudholt@sheffield.ac.uk
http://staffwww.dcs.shef.ac.uk/~dirk/campus_only/com1006/

Based on Chapter 3 in Clements, Principles of Computer Hardware

Aims of this lecture

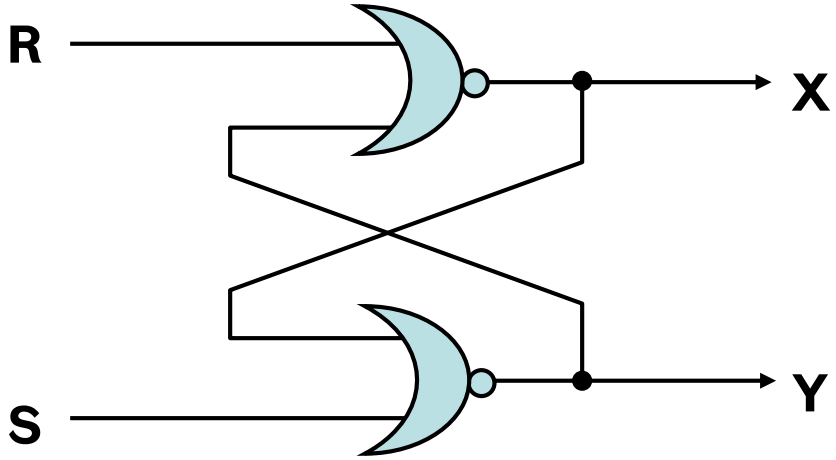
- To show how simple memory elements are implemented in digital circuits using flip-flops.
- To introduce the major kinds of flip-flops, demonstrate their behaviour and illustrate their applications.
- To introduce the concept of clocking.
- To compare level, edge and master-slave triggering.
- To show how sequential circuits can be designed with flip-flops and combinatorial logic.
- To show how flip-flops can be used to implement more complicated circuits such as shift registers and counters.

Sequential logic

- So far we have built logic circuits using **combinatorial elements** whose outputs are a function of their inputs only.
- Now we consider **sequential circuits** whose outputs depends on their current inputs and their previous inputs.
- Example: a counter
- **Flip-flops** (also called bistables) are the building blocks of sequential circuits.
- The output of a flip-flop can remain in one of two states indefinitely, even if the input changes.
- Flip-flops are therefore basic memory elements.

The RS flip-flop

- The RS flip-flop is the simplest flip-flop, and consists of two **cross-coupled** NOR gates.
- We have two inputs R and S, and two outputs X and Y
- Recall the truth table for a NOR gate $\overline{P+Q}$, shown right.



P	Q	$F = \overline{P+Q}$
0	0	1
0	1	0
1	0	0
1	1	0

Analysis of the RS flip-flop

- We can write expressions for the outputs X and Y in terms of the inputs R and S:

  $X = \overline{R+Y}$ $\qquad\qquad$ $Y = \overline{S+X}$

- Substituting for Y into the expression for X,

  $$X = \overline{R + (\overline{S+X})}$$

  $$= \overline{R} \cdot (\overline{\overline{S + X}}) \qquad \text{by de Morgan's theorem}$$

  $$= \overline{R} \cdot (S + X) = \overline{R}\,S + \overline{R}\,X$$

- What does $X = \overline{R}\,S + \overline{R}\,X$ actually mean? The X on the RHS is the old value of X, and the X on the LHS is the new value of X. To make this clear, we can write $X^{+} = \overline{R}\,S + \overline{R}\,X$

Truth table for the RS flip-flop

- Recall $X^+ = \overline{R}\,S + \overline{R}\,X$
- If R=1, the output X^+ becomes 0 regardless of its previous value, and stays 0 when R=0
- **R=1 causes the output to reset (become 0)**
- If S=1 and R=0, the output X^+ becomes 1 regardless of its previous value, and stays 1 when S=0
- **S=1 causes the output to set (become 1)**

Inputs			Output
R	S	X	X^+
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

Set and reset in the RS flip-flop

- RS flip-flops are often drawn as a "box" with inputs:
  - R for reset
  - S for set

- Historically the output X is labelled Q.
- Checking the original circuit, we can see that except for the case R=S=1, X and Y are complements. Therefore we can label the other output $\overline{Q}$.

❓ **Verify this by writing the truth table for** $Y=\overline{S+X^{+}}$

A revised truth table for the RS flip-flop

- Note that the case R=S=1 is **forbidden**, since it implies $Q=\overline{Q}=0$
- In practice the input R=S=1 is simply avoided.

Inputs			Output	
R	S	Q	Q^+	
0	0	0	0	No change
0	0	1	1	No change
0	1	0	1	Set
0	1	1	1	Set
1	0	0	0	Reset
1	0	1	0	Reset
1	1	0	0	Forbidden
1	1	1	0	Forbidden

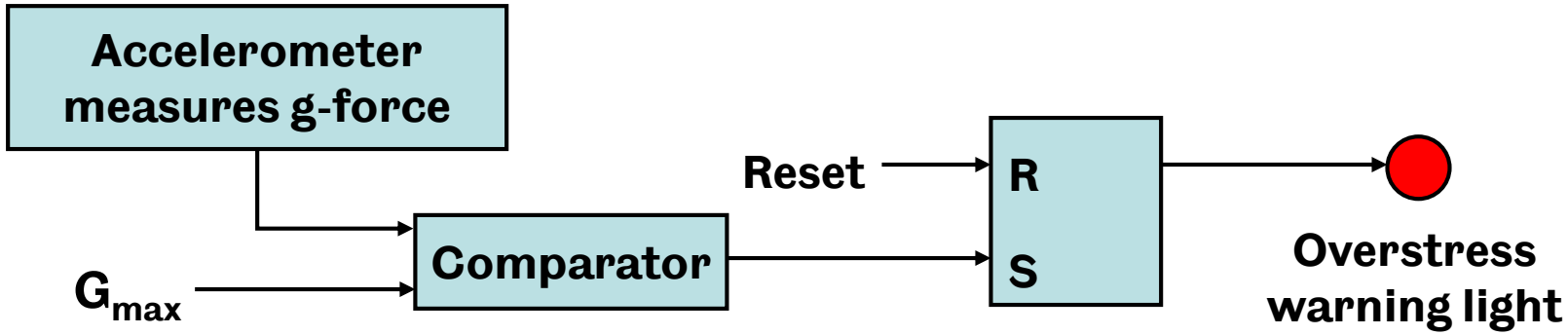
Algebraic RS flip-flop truth table

- A form of the truth table that uses the algebraic value of Q is frequently used, because it is more concise:

Inputs		Output	Description
R	S	Q^+	
0	0	Q	No change
0	1	1	Set output to 1
1	0	0	Reset output to 0
1	1	X	Forbidden

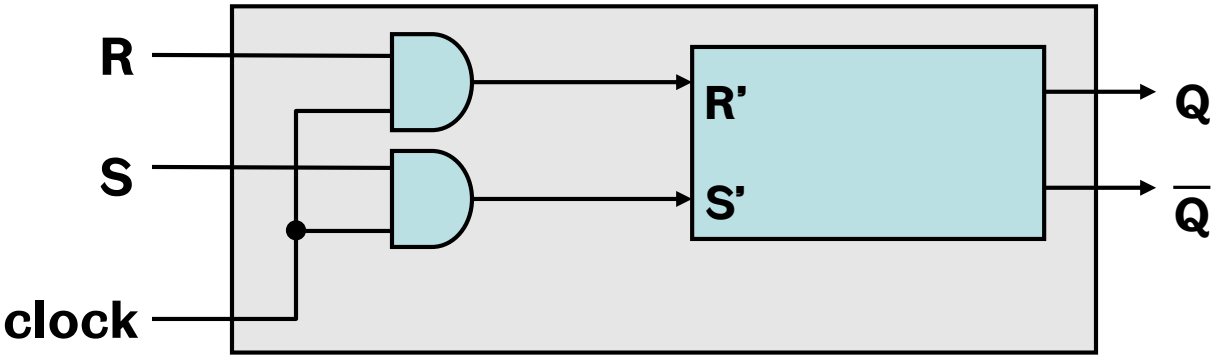
Application of the RS flip-flop

- An important application of the RS flip-flop is the recording of brief events (i.e., it acts as a **latch**).
- Example: it is important to record situations in which an aircraft has been placed under high stress, even if briefly.
- This circuit uses a RS flip-flop to set a warning light if the g-force measured by an accelerometer exceeds a limit:



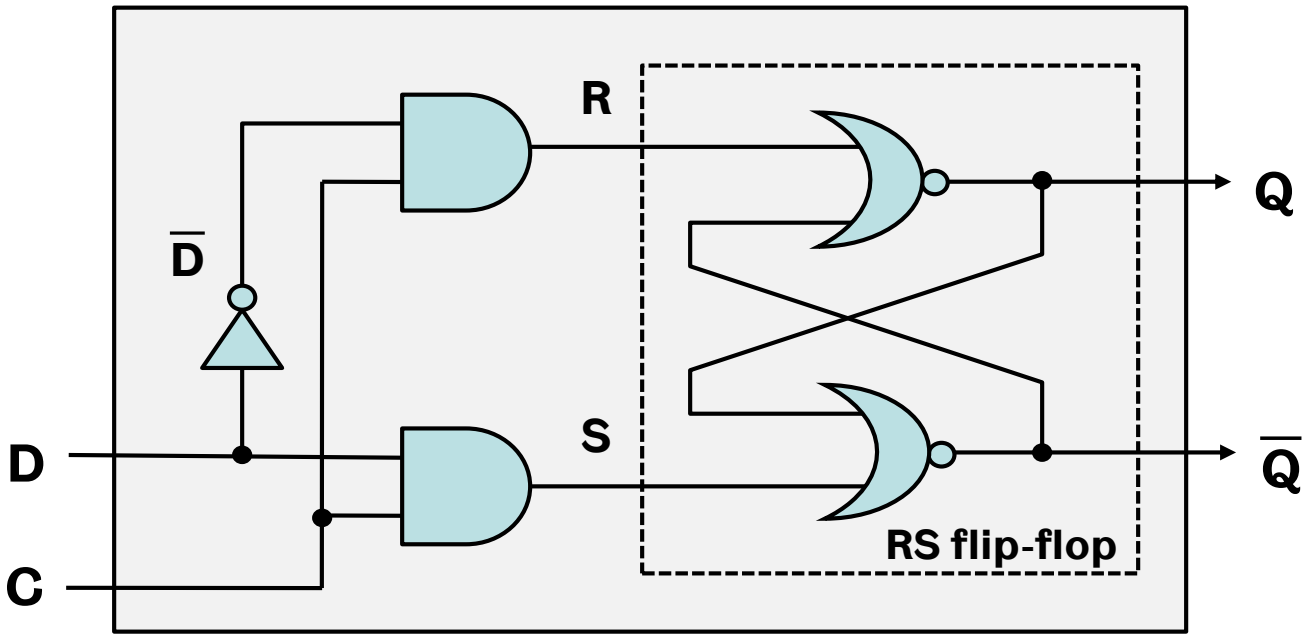
Clocked flip-flops

- In a **clocked** flip-flop, inputs are ignored while the clock signal is 0. The clock input prevents the flip-flop from acting until the right time (useful for dealing with **glitches**).
- We can make a **clocked RS-flip flop** by ANDing the R and S inputs with a clock signal:



- clock=0 means R' = S' = 0, ignoring R and S.
  clock=1 forwards inputs signals: R' = R and S' = S.

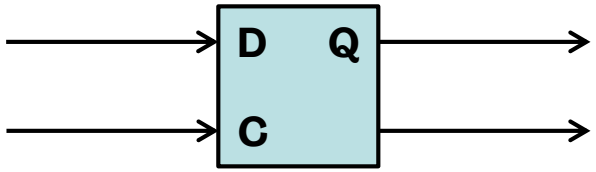
The D flip-flop (D for Data / Delay)



- The D flip-flop consists of an RS flip-flop and a few gates.
- It is a clocked flip-flop, with data (D) and clock (C) inputs.
- When clocked, RS flip-flop is either reset (D=0) or set (D=1). This avoids R=S=1.

Behaviour of the D flip-flop

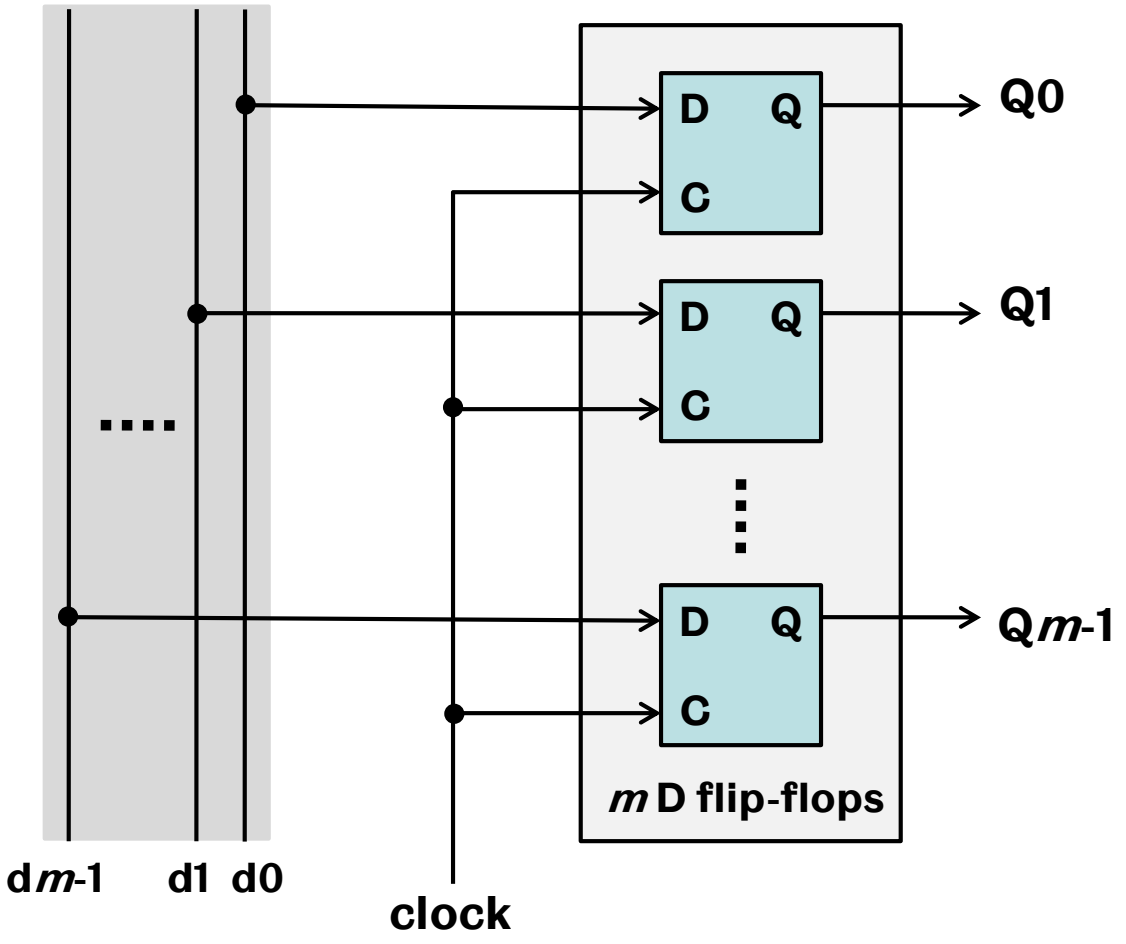
- When a D flip-flop is clocked, the D input is transferred to the output and remains constant until the next time it is clocked.
- The D flip-flop is suitable for implementing simple memory elements; it records the state of the input and remembers it until clocked again.



Inputs		Output
C	D	Q^+
0	0	Q
0	1	Q
1	0	0
1	1	1

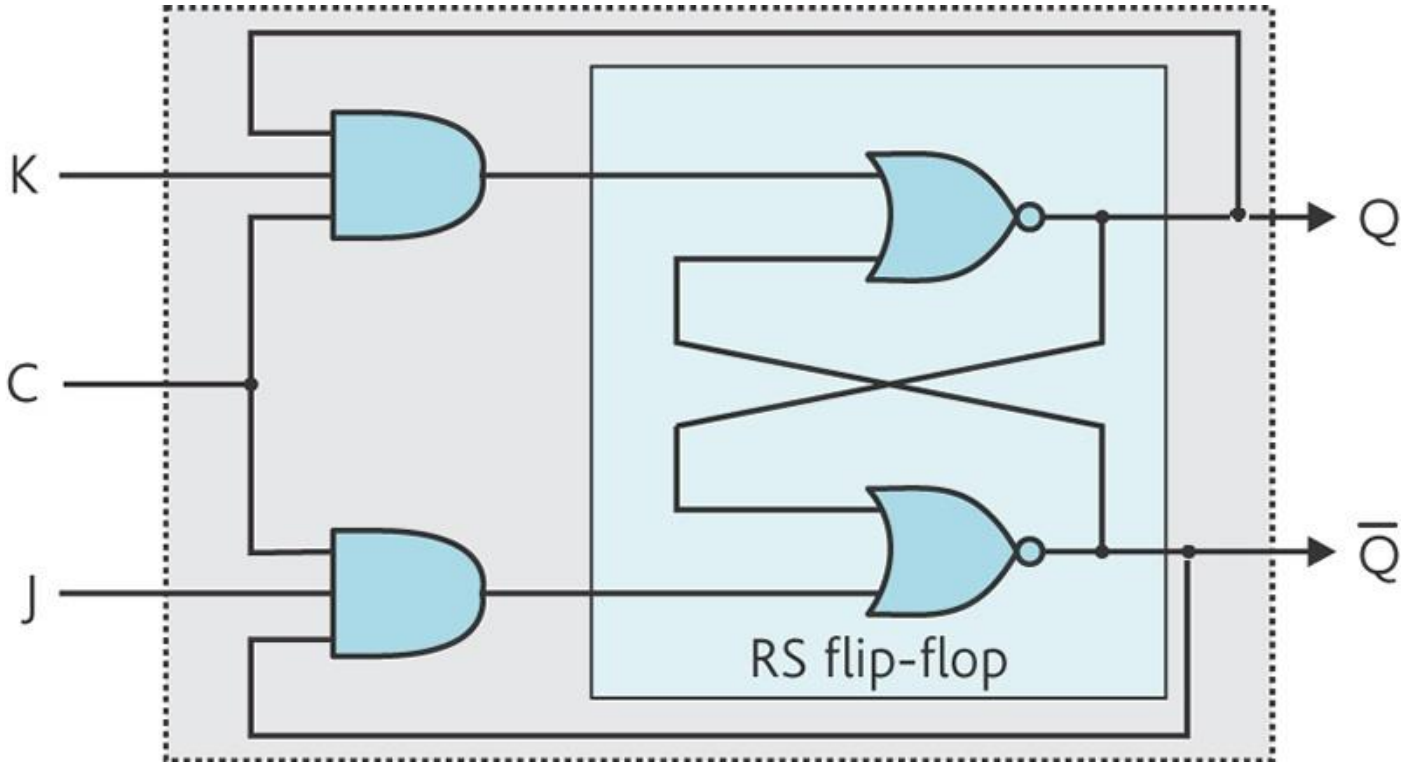
Register implemented with D flip-flops

- An *m*-bit wide bus transfers data into a register consisting of *m* D flip-flops.
- Bus contents are ignored when clock=0. Data on all *m* bus lines are stored when clock=1.



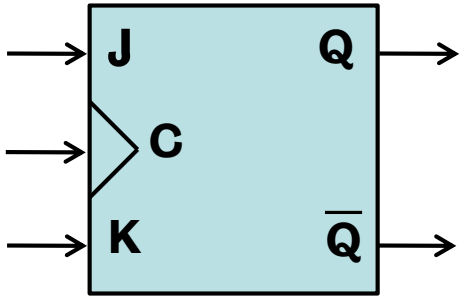
JK flip-flop

- Like RS flip-flop, but with additional logic avoiding R=S=1.
- Allows to maintain, set, clear, and toggle current state.



Truth table for JK flip-flop

- JK flip-flop behaves like an RS flip-flop for all values of J (set) and K (clear) except J=K=1.
- When J=K=1 the output of the JK flip-flop changes state (**toggles**) each time it is clocked.



Inputs		Output	Description
J	K	Q^+	
0	0	Q	No change
0	1	0	Clear ("**K**lear")
1	0	1	Set
1	1	$\overline{Q}$	Toggle

Summary of flip-flop types

- Flip-flops have internal states as well as external inputs, and therefore function as memory elements
- **RS flip-flop**. Two inputs, (R)eset and (S)et. R=S=1 is forbidden. It is a latch, used for recording (even transient) events.
- **D flip-flop**. Two inputs, (D)ata and (C)lock. It records the state of D and holds it constant until C is clocked. It is used for memory elements such as registers.
- **JK flip-flop**. Has three inputs, J (Set), K (Klear) and (C)lock. Output remains in previous state so long as it is not clocked. Works like RS flip-flop, but toggles if J=K=1.

Clocking – level triggered

- There are actually three ways of clocking a flip-flop; **level**, **edge** and **master-slave** triggering.
- A level-sensitive flip-flop is triggered whenever the clock is in a particular logical state (0 or 1; we assume 1 hereafter).
- The clocked RS flip-flop on slide 11 is level triggered because it only responds to the R and S inputs when the clock is input is high.
- Level-triggering is unsuited to some applications (e.g., digital circuits in which the output is fed back into the input, since the state of the circuit will oscillate during a single clock pulse).

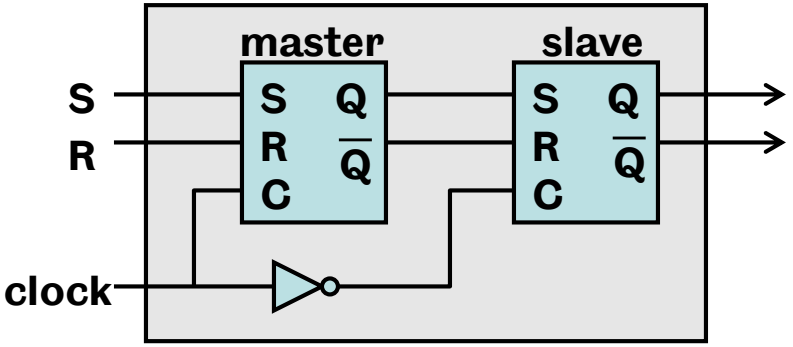
Clocking – edge triggered

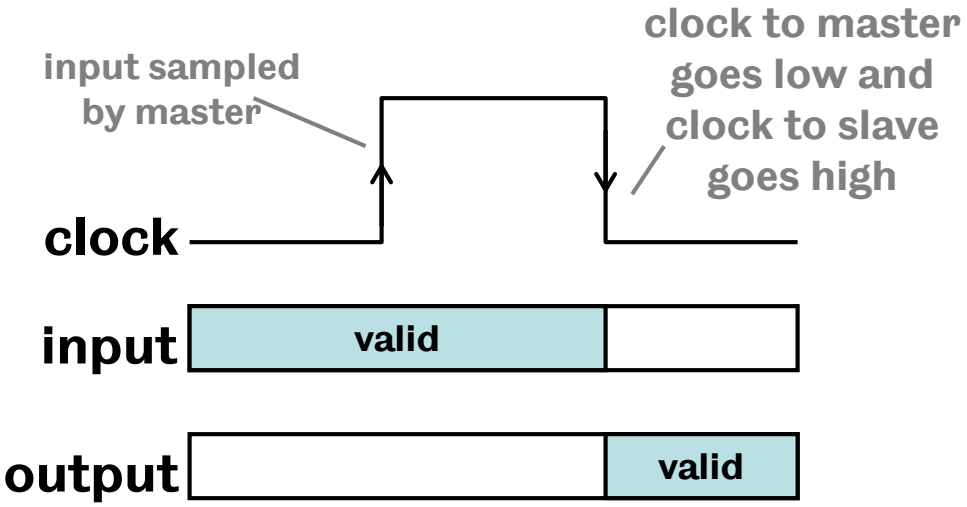
- Edge-triggered flip-flops are clocked by the transition of the clock from zero to one (**rising edge**), or one to zero (**falling edge**).

- Edge-triggered flip-flops solve many of the problems of level triggering because the transition is very short (less than 1ns).
- Clock skew (tiny variations in arrival time of pulses at each clock input) can cause a problem for edge triggering.

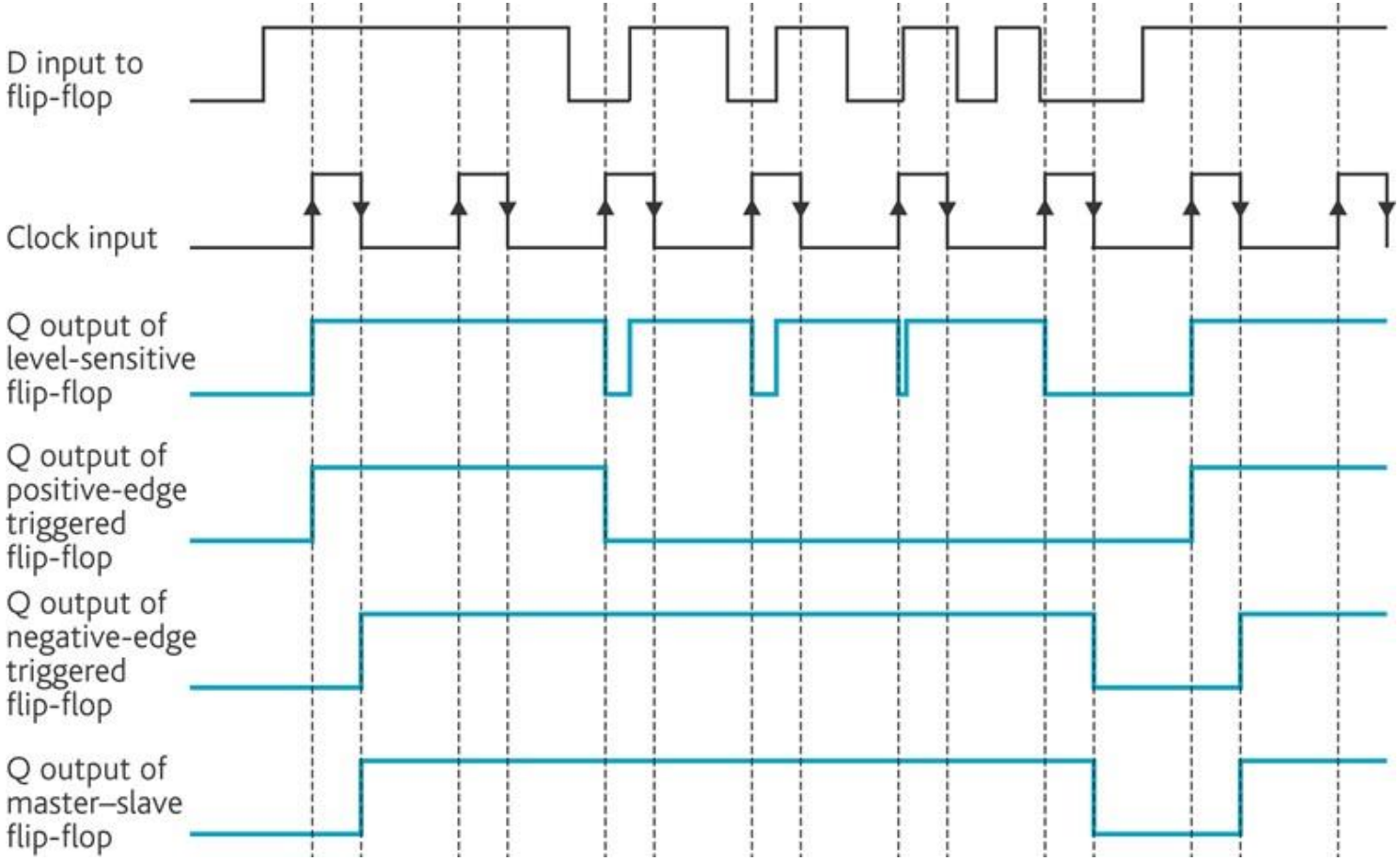
Clocking - master/slave

- A master-slave (MS) flip-flop is actually two flip-flops operating in series. The master flip-flop captures the input, the slave flip-flop holds the output.
- Master triggers on high level, slave on low level.
- Like an air-lock in a submarine!



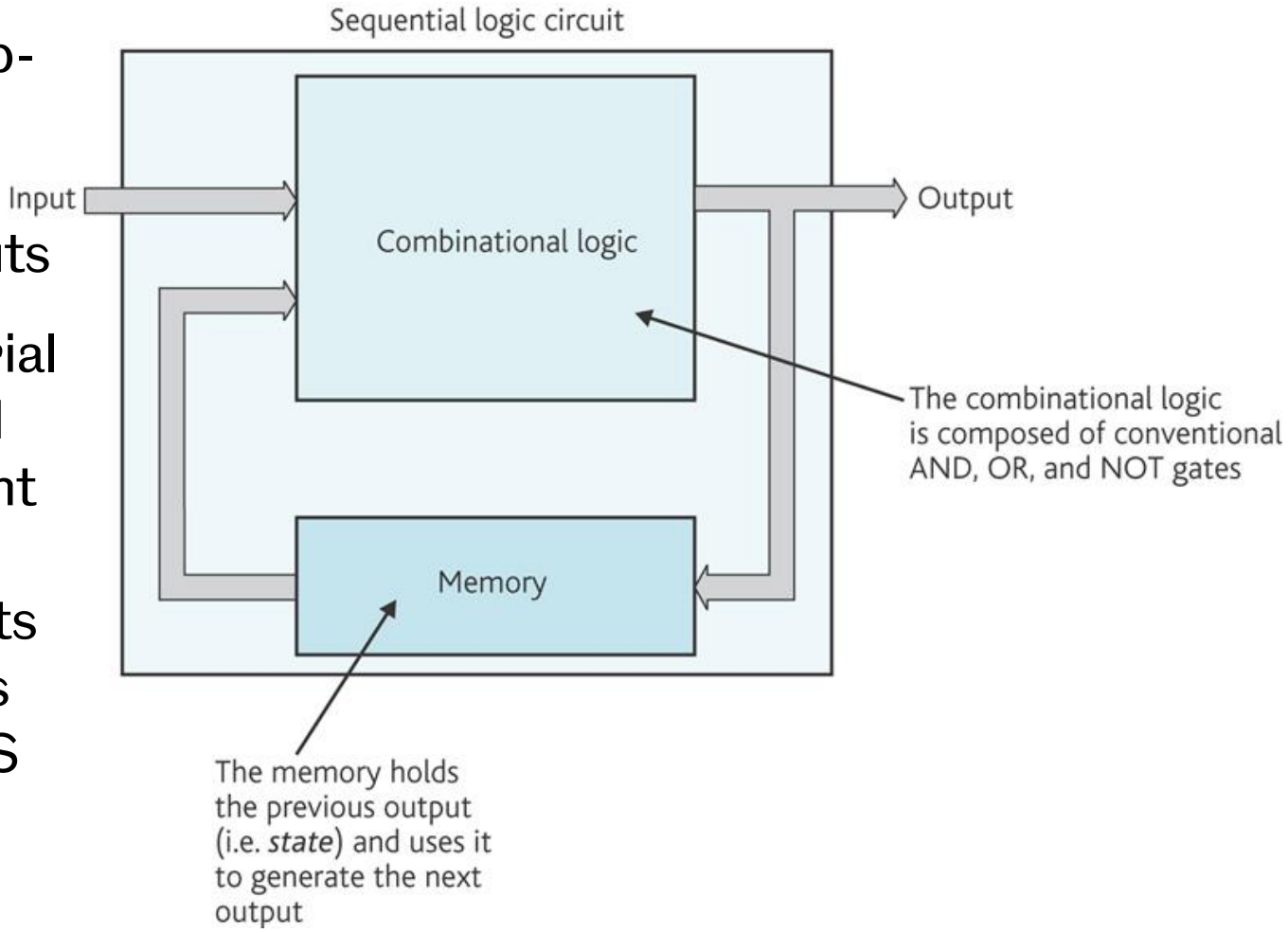


Comparison of triggering behaviour



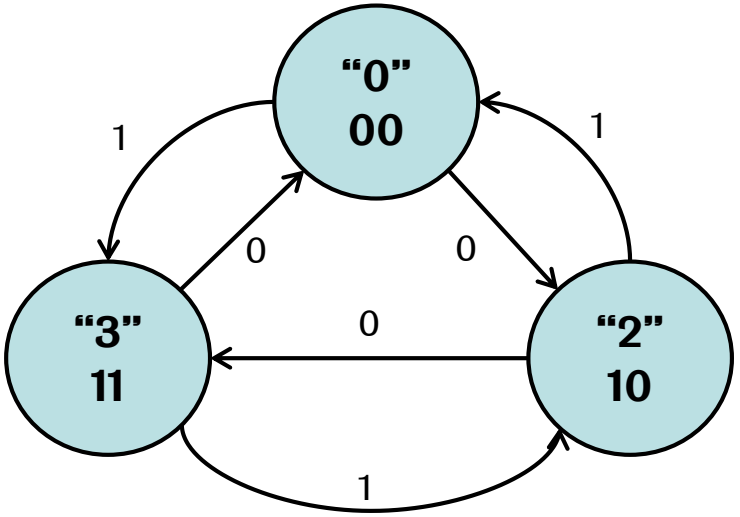
How to design sequential circuits

- States of flip-flops can be treated like regular inputs
- Combinatorial logic is used to implement outputs as well as inputs for flip-flops (J&K or R&S or D).



Example: custom up-and-down counter

- Let's design a two-bit counter that counts 0, 2, 3 (skipping 1).
- It has a switch, an input A, determining whether to count up or down when clocked.
  - If A=0, the counter counts up
  - If A=1, the counter counts down
- Example of a **state machine** used to model sequential circuits. (loads more of this in COM2003)
- The counter never reaches „01".



- Need two flip-flops to represent current number/state Q_1Q_0. Any flip-flop type will do – let's choose to use two JK flip-flops.

Step 1: truth table for state transitions

- Figure out how the next states need to be set, according to inputs and current state.

Input	Current state		Next state	
A	Q_1	Q_0	Q_1^+	Q_0^+
0	0	0	1	0
0	0	1	X	X
0	1	0	1	1
0	1	1	0	0
1	0	0	1	1
1	0	1	X	X
1	1	0	0	0
1	1	1	1	0

Step 2: determine inputs for flip-flops

- New state Q_1^+, Q_0^+ being determined by flip-flop inputs J_1, K_1, J_0, K_0.
- Work out how to set these to get the desired next state.
- Use „don't cares" if next state is „X".

Input	Current state		Next state		Inputs for JK-flip flops			
A	Q_1	Q_0	Q_1^+	Q_0^+	J_1	K_1	J_0	K_0
0	0	0	1	0				
0	0	1	X	X	X	X	X	X
0	1	0	1	1				
0	1	1	0	0				
1	0	0	1	1				
1	0	1	X	X	X	X	X	X
1	1	0	0	0				
1	1	1	1	0				

How to set inputs for flip-flops

- Force next state to 0 or 1 according to flip-flop‘s truth table.
- Depending on current state, there might be further options. (Recall: the more „don‘t cares“, the better!)

Curr. State	Next State	RS inputs	
Q	Q+	R	S
0	0	X	0
0	1	0	1
1	0	1	0
1	1	0	X

avoid R=S=1!

Curr. State	Next State	JK inputs	
Q	Q+	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

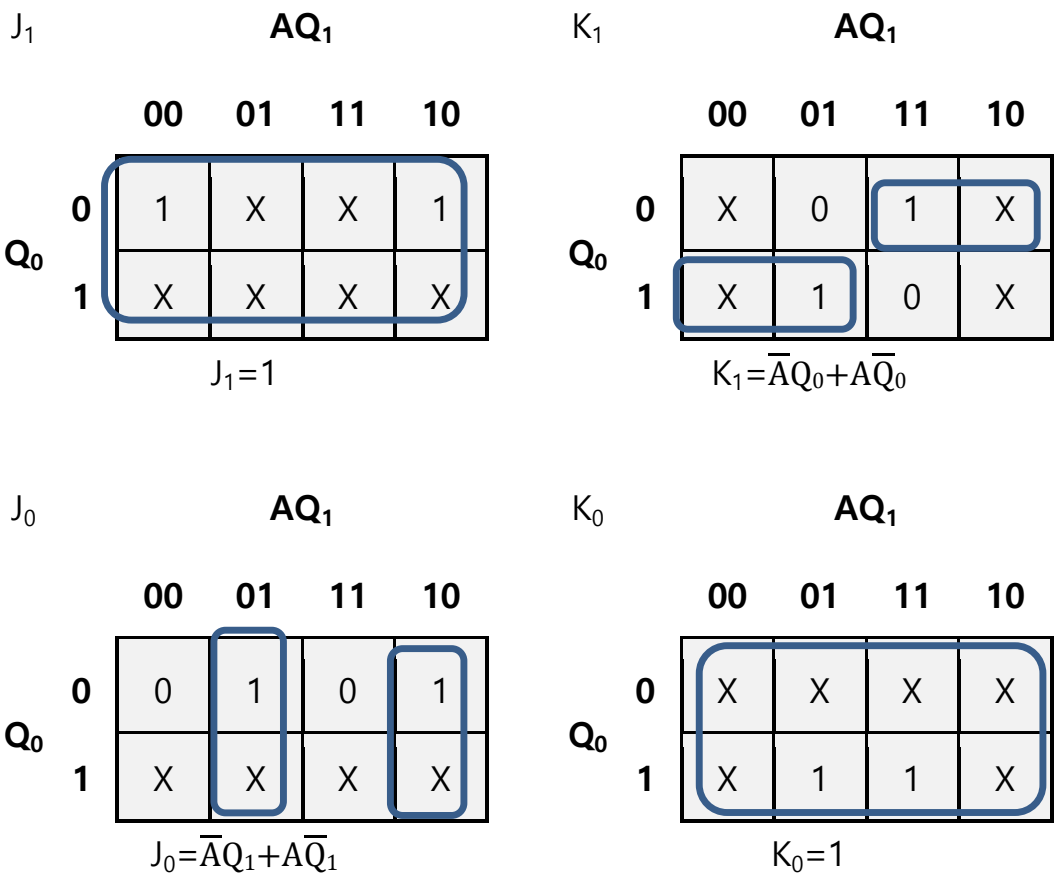
Curr. State	Next State	D input
Q	Q+	D
0	0	0
0	1	1
1	0	0
1	1	1

Step 2: determine inputs for flip-flops

- Here's the completed table:

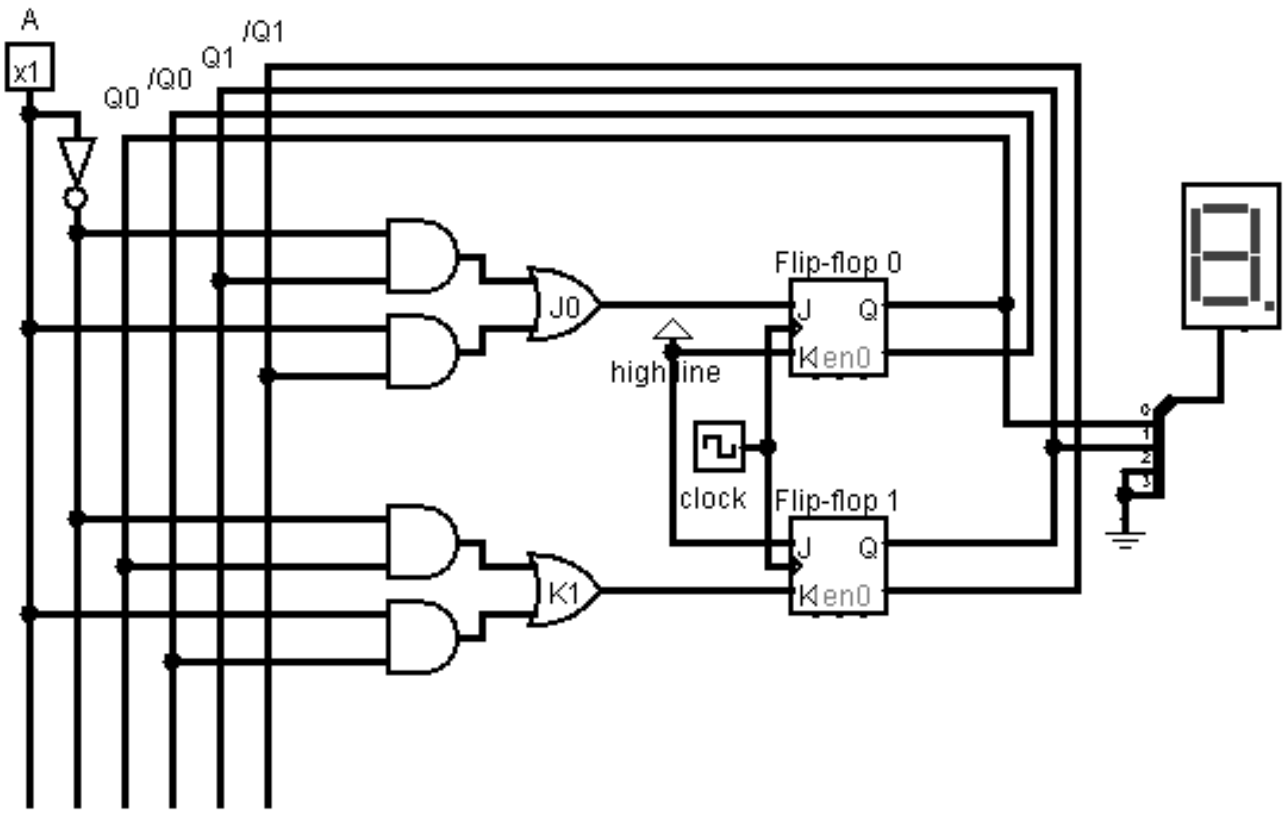
Input	Current state		Next state		Inputs for JK-flip flops			
A	Q_1	Q_0	Q_1^+	Q_0^+	J_1	K_1	J_0	K_0
0	0	0	1	0	1	X	0	X
0	0	1	X	X	X	X	X	X
0	1	0	1	1	X	0	1	X
0	1	1	0	0	X	1	X	1
1	0	0	1	1	1	X	1	X
1	0	1	X	X	X	X	X	X
1	1	0	0	0	X	1	0	X
1	1	1	1	0	X	0	X	1

Step 3: design circuits for flip-flop inputs



Step 4: assemble circuits and flip-flops

- Q_0, Q_1, $\overline{Q_0}$, $\overline{Q_1}$ used as inputs to circuits for J_0, K_0, J_1, K_1
- Here: J_1=K_0=1 implemented by line to power
- Clock with edge triggering.
- Added 7-segment display for output.



Synchronous & asynchronous systems

- **Synchronous system** – a clocked system in which all processes share the same clock signal. All outputs are held constant until the next time the flip-flops are clocked.
- **Asynchronous system** – the output of one process triggers the start of the next.
- Previous design method is for **synchronous systems**.
- Asynchronous systems are more complex and harder to design, but faster and use less power.
- Synchronous systems are often required due to the effect of propagation delays on digital circuits.

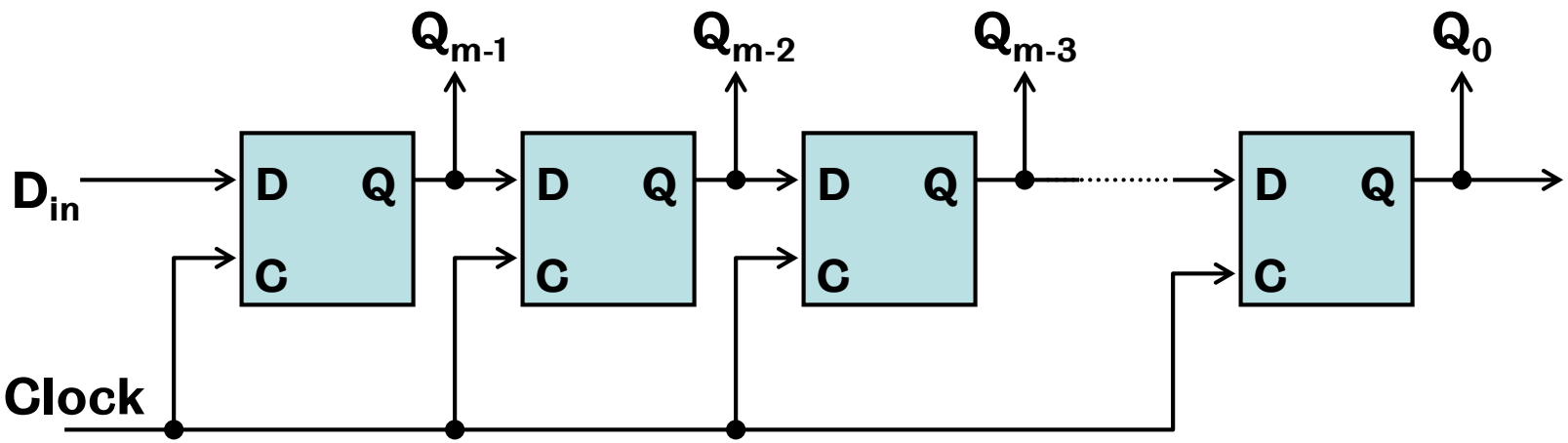
Shift register

- By modifying the circuit for a register we can build a **shift register** in which the bits are moved one place right each time the register is clocked, e.g.

`01110101`	original contents
`00111010`	after clocking once
`00011101`	after clocking twice

- The right-most bit is lost and a zero is shifted in the left.
- If the bit pattern represents an unsigned binary number, shifting it right has the effect of **dividing** the number **by two**.

Right shift register with D flip-flops

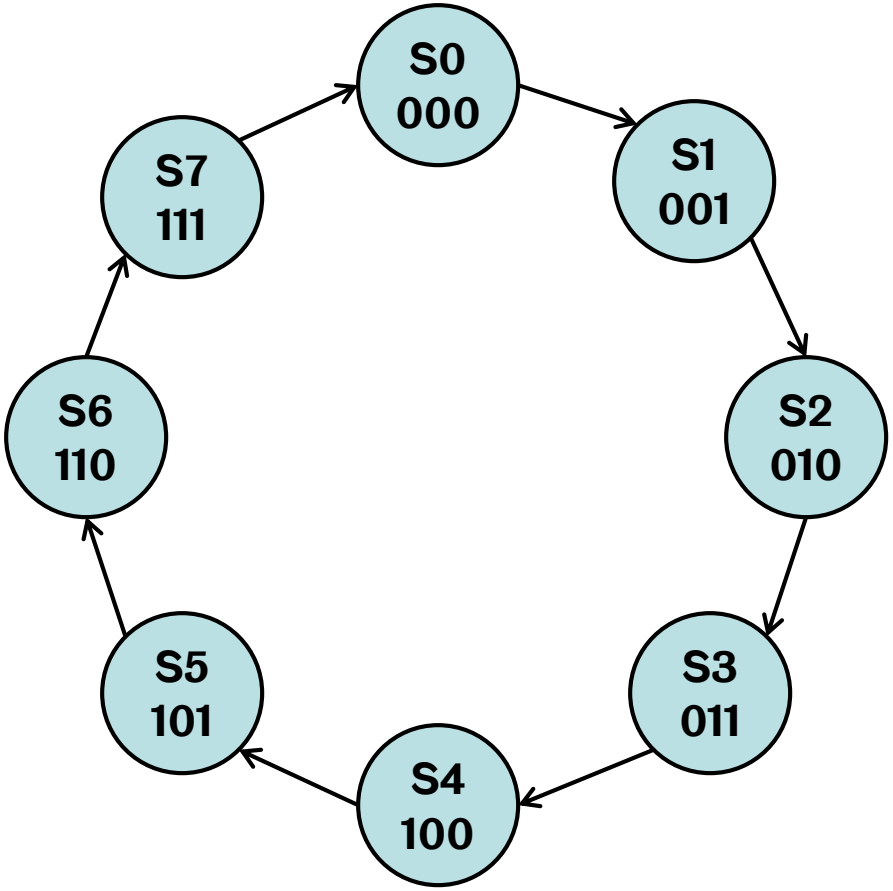


- All clock inputs are connected together so that the D flip-flops are clocked simultaneously.
- On each clock pulse the output (Q) of each flip-flop is copied to the input (D) of the next stage on the right.

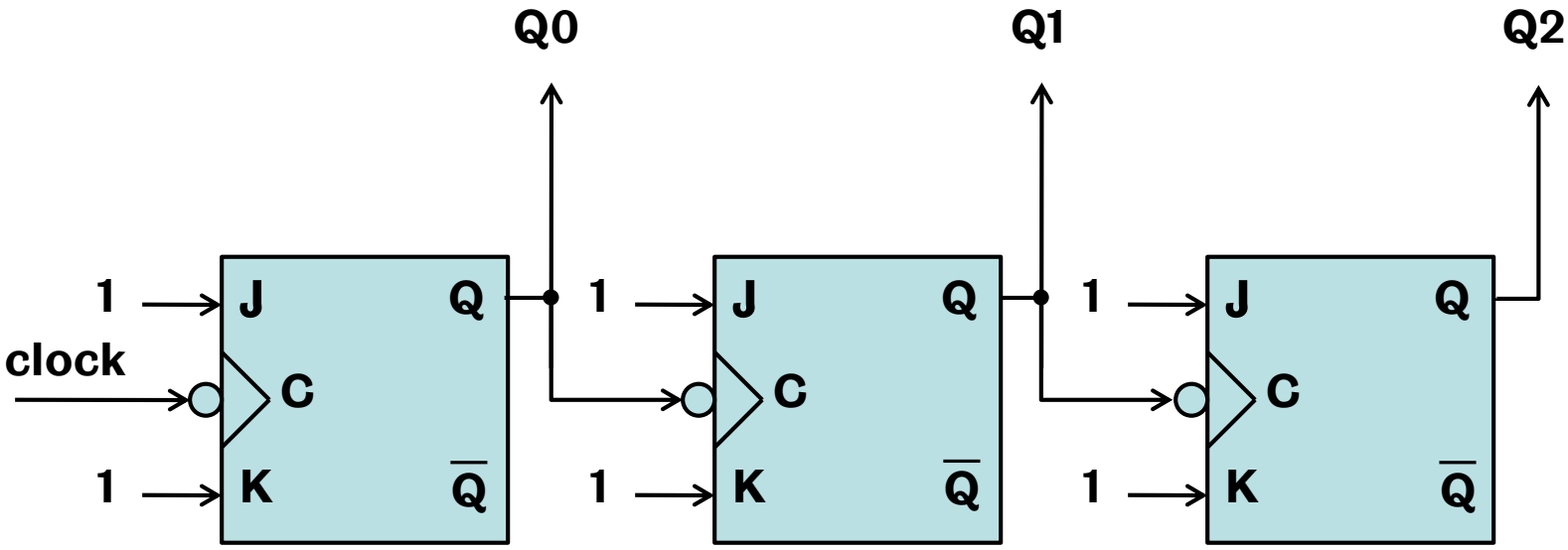
❓ **Should the flip-flops be level or edge triggered?**

Binary up-counter

- Another example of a **state machine** to model sequential circuits.
- Implementing an ordinary 3-bit binary up-counter using a simple state machine.
- States S0-S7, each outputs the current count in the sequence.

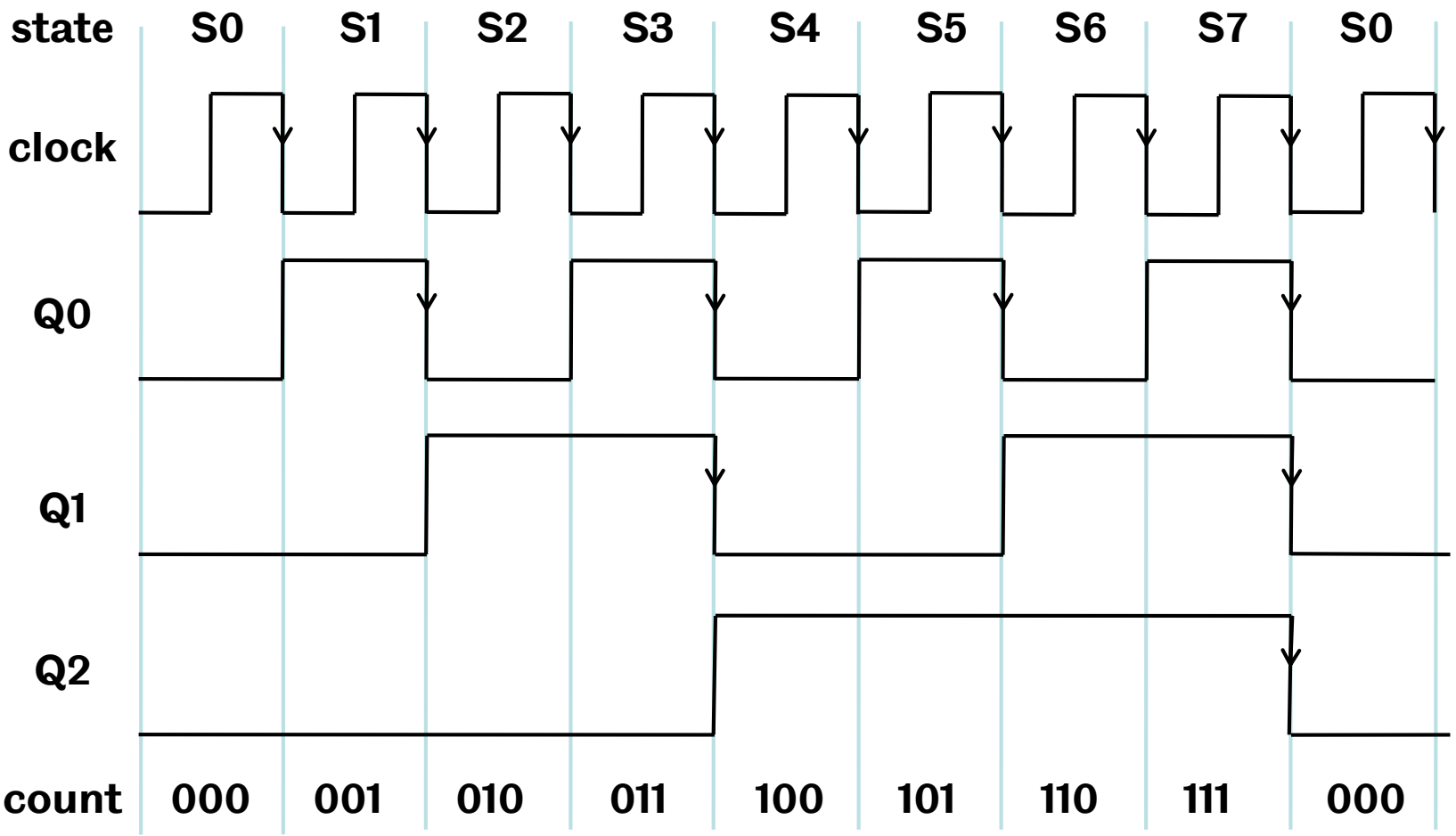


Asynchronous binary up-counter



- J and K inputs of each JK flip-flop are connected to logical 1 so that the outputs **toggle** on each clock pulse.
- Flip-flops are negative-edge triggered, change state on falling edge of the clock pulse (indicated by the circle).
- Each output triggers the next input (a **ripple counter**).

Timing diagram for 3-bit up counter



Summary

- Flip-flops are basic memory elements.
- Different kinds of flip-flop (D, RS, JK) have different applications. JK flip-flops are the most general kind.
- Digital circuits can operate synchronously or asynchronously.
- In synchronous (clocked) circuits, the clocking can be level, edge or master-slave triggered.
- Synchronous sequential circuits can be designed with flip-flops and combinatorial logic controlling their inputs to implement the desired state transition.
- Flip-flops can be combined to make registers and shift registers.
- JK flip-flops can be used to implement simple counters.