

Digital Electronics 3: Sequential Logic

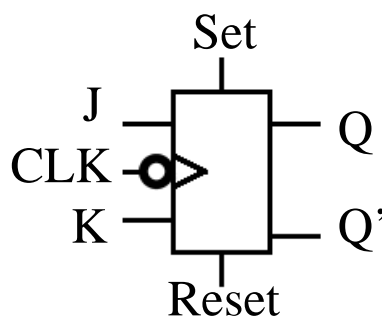
This is the third and final digital electronics lecture. A memory element is perhaps not the most dynamic application. In the lab you will build a **device to count** and a circuit for **controlling a motor**. Here we will introduce the relevant background material.

Sequential Logic: Asynchronous and Synchronous applications

The names **asynchronous** and **synchronous** relate to the two ways that a clock can be used in a digital circuit. The most primitive is asynchronous, which we will look at first.

Opaque latch: from JK Flip-Flop to D-Latch (H&H, 8.24, p.523)

Last lecture we went through numerous flip flop circuits, ending up with the JK Flip-Flop. We skipped over the precise details and finished up with a symbol to represent this final type of flip flop. As a reminder here are its **circuit symbol** and its **operations table** again:



J	K	Q_{n+1}
0	0	Q_n
0	1	0
1	0	1
1	1	Q_n'

You are going to be using these in the lab as part of checkpoint D2.

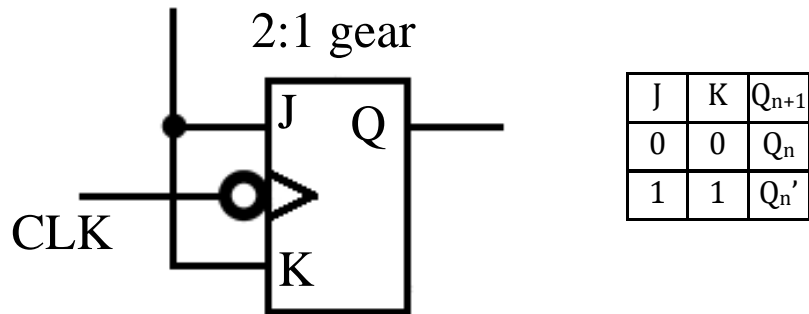
The operation table looks like a truth table, however, there is an important difference: the values in the operation table are only true if there is a **clock** signal attached to the input labelled "CLK". The output updates from Q_n to Q_{n+1} at the selected edge of the clock signal. The exaggerated *circle in front of the CLK* input of the JK Flip-Flop implies a NOT. This means that the output updates at the **falling edge**, where the clock signal drops from a 1 to a 0.

The JK Flip-Flop is typically used in one of two ways:

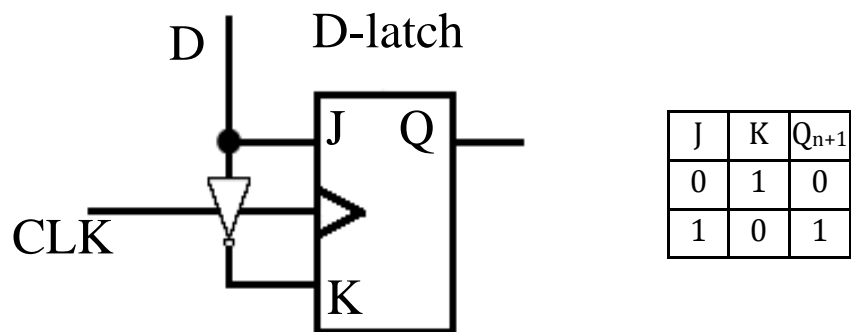
- either the **two inputs** are forced to always be the **same** (i.e. yielding the memory and alternation setting of the operations table)
- or the **two inputs** are forced to always be **opposite** (i.e. yielding the set and reset functions of the operations table).

The two cases are discussed below.

The case with the *two inputs* forced to be the *same* is most typically used in **counting applications**. If $J=K=1$ then at every time the clock drops the output switches state (from 1 to 0 or vice versa). Since the clock only drops once per complete clock cycle this leads to a series of pulses at the output of the JK Flip-Flop which are *twice as long as those from the clock*, it behaves like a **2:1 gear**. Here are the circuit and the operations table:



The alternative, where the *two inputs* forced to be the *opposite*, is called a **D-latch**. In this case, every time the clock drops the value at the output is updated with the value at the input. This is useful in *controlling when information is available*. For example, this is used converting information **from serial into parallel** form.



You may have noticed that the D-latch lacks the circle at the clock input. Thus, it acts on the **rising edge** of the clock, where the clock signal raises from a 0 to a 1. This way the *data flow is effectively controlled* without using up more clock cycles.

Again – these operation tables are what you observe when you have *attached a clock signal* to the input labelled CLK. Without this square wave made up of 1s and 0s nothing will happen.

Counting: (H&H, 8.17, p.511-2)

Counting in binary can easily be identified as a series of frequency halving steps. Here we look at 4 bit numbers again. The four columns on the right at the 8s, 4s, 2s and 1s of the binary number system. The cells containing the bit value 1 are shaded grey, so that the pattern stands out.

Decimal	2^3	2^2	2^1	2^0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

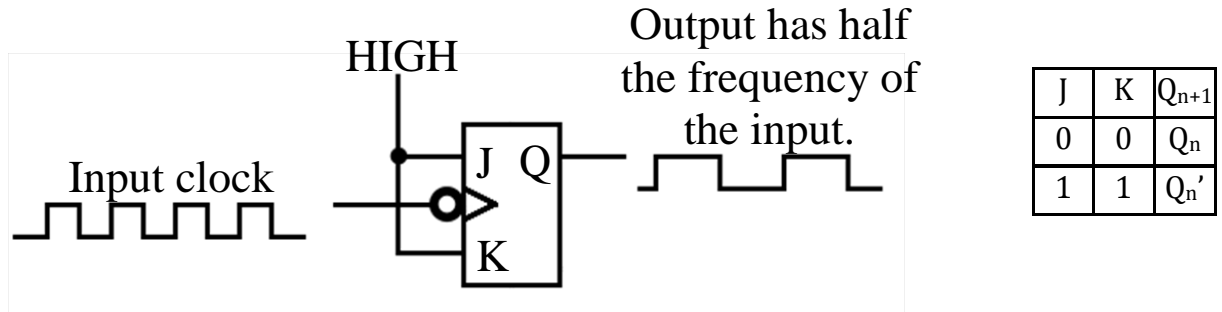
As you can see the frequency halves as you go from units, to twos. It halves again from twos to fours and again from fours to eights.

The process of counting is to line up a series of oscillators each one oscillating half as fast as the one before. This asks for the 2:1 gear configuration of the JK Flip-Flop, with J=K=1. A cascade of these, successively feeding through the discounted clock signal, is called a **ripple counter**. The number of JK Flip-Flop units in the cascade, i.e. bits, determines the maximum size of counts which can be represented.

You are now supposed to understand that counting involves changes in the frequency of occurrence of 1s.

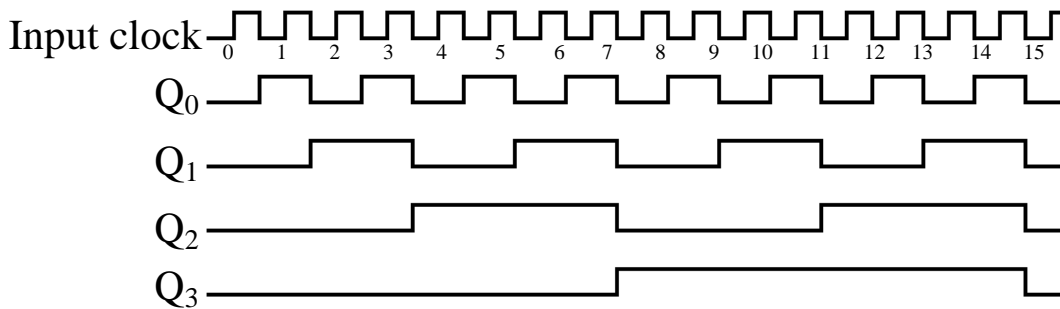
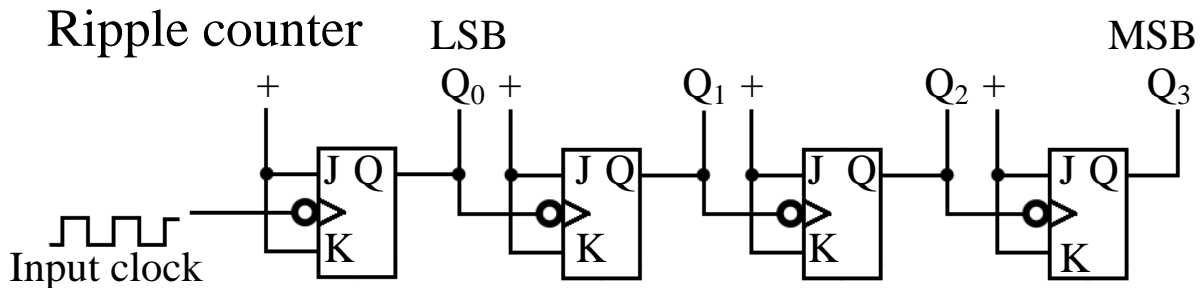
Ripple counter:

Using a JK Flip-Flop, configured with J=K=1, the output changes state on the falling edge of every clock pulse. Hence there will now be one output clock cycle for every two input clock cycle. This gives the frequency halving which is needed to count. This effect is pictured below.



To make the full counter several of these units will need to be cascaded, one for each bit of the binary number to be represented. The output of each stage will become the clock for the next stage. Such a circuit design is called **asynchronous**. This means that different parts of the circuit *see different clock signals*.

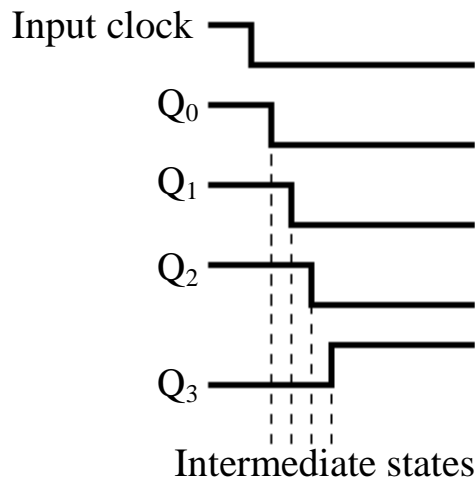
The cascaded stages of the counter are shown below. The “+” symbols indicate connections to a logic 1 setting (a positive voltage). The output with the *highest frequency*, Q₀, is the **least significant bit** (LSB). The output with the *lowest frequency*, Q₃, gives the **most significant bit** (MSB). All outputs together also are referred to as **n-bit register**. The output of each stage feeds into the clock input of the next stage and the relative behaviour of the clock and the output gates is given in a **timing diagram** below the circuit. In such diagrams time flows to the right and synchronous events are drawn above each other.



In the style becoming characteristic for this course – we are now looking at why that circuit is not such a great idea.

Ripple counter problems:

Problems develop because the elements are **not synchronised** to the same clock. As the clock signal propagates *delays build up* in the ripple counter circuit. Since the different clock signals arrive at different flip flops at slightly different times they update their output at slightly different times. Hence the status of the output register goes through *bogus intermediate states* on its transition from one state to another.



We can write the numbers represented in this illustration in brackets (Q₃ Q₂ Q₁ Q₀). The intention is to go from the left state, 7 (0111), to the right state, 8 (1000). But as during signal propagation through the Ripple Counter the output bits Q₀, Q₁, Q₂ and Q₃ update sequentially, the values represented in the register passes through the intermediate states: 6 (0110), 4 (0100) and 0 (0000), before the final state is reached. Indeed, the sequence 7, 6, 4, 0, 8 is not normally considered helpful in counting from 7 to 8!

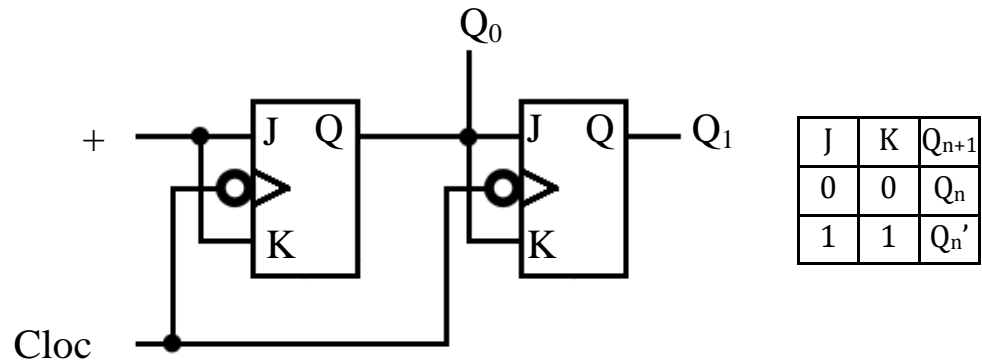
The delays, as before, stem from the underlying analogue nature of digital electronics. **Signal propagation** along tracks and cables is limited (typically 5ns/m) by the speed of light in these media. State transitions are limited by the **slew rate** by which voltage states can change (typically 1000V/μs). As a result, for packaged units one should expect delays of up to 10ns (depending on the complexity of the internal circuit) between input and output. Therefore a Ripple Counter may 'clock up' (☺) a significant delay between LSB and MSB.

The solution is to have every part of the counter controlled by the same clock. Then each flip flop changes state at the same time – which minimizes errors. Such a circuit, where every component *shares the same clock* is called **synchronous**.

[You are now supposed to understand how to make an asynchronous counter from JK flip flops and why this design has problems with delays.](#)

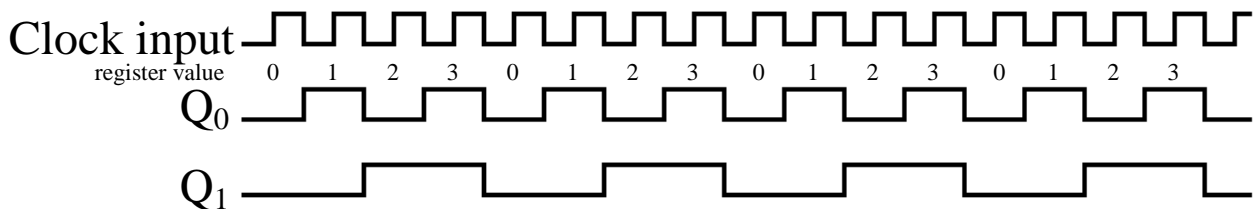
Synchronous counter:

Instead of having the clock trigger the first gate only, here the **same clock triggers every gate** directly. This reduces the problem of propagation delays to that of a sufficiently even distribution of the clock signal. The important design change is that now the Q output of the first gate *actually modifies the JK values* of the second flip flop. While in the previous circuit $J=K=1$ was always maintained.



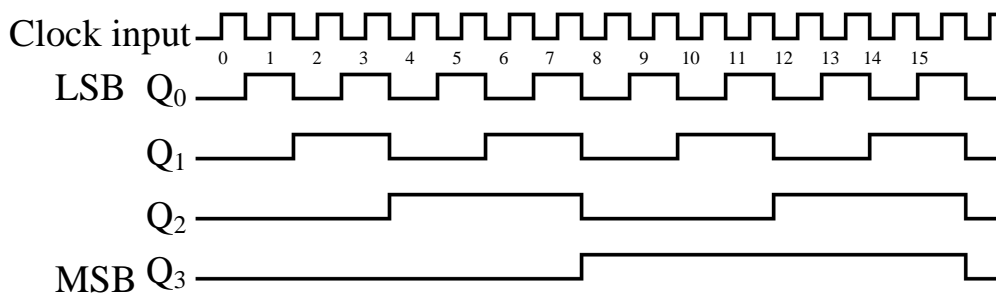
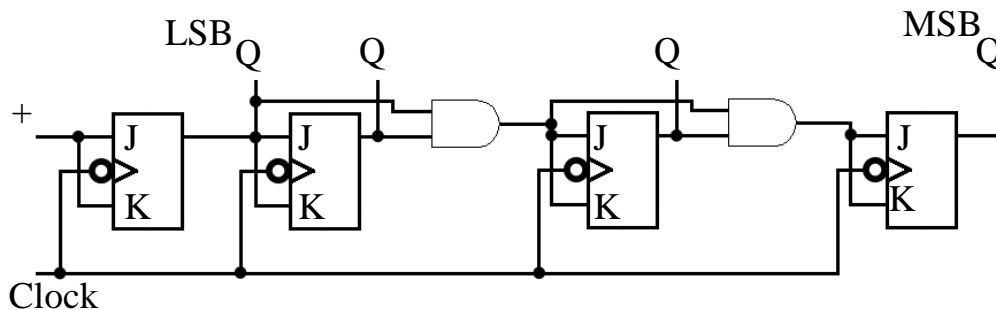
Due to the characteristics of the operations table of the JK Flip-Flop the 2:1 gear functionality is still preserved: Q_0 toggles with every clock pulse, Q_1 only toggles when Q_0 is high.

A counter with **no delays** – the states are updating at the same time, i.e. without the nasty intermediate states in the transitions of the clock. If you look on the 10ns timescale at the points at which the numbers are changing you will find that they *flip simultaneously*.



You are now supposed to understand how to make a 2-bit synchronous counting circuit.

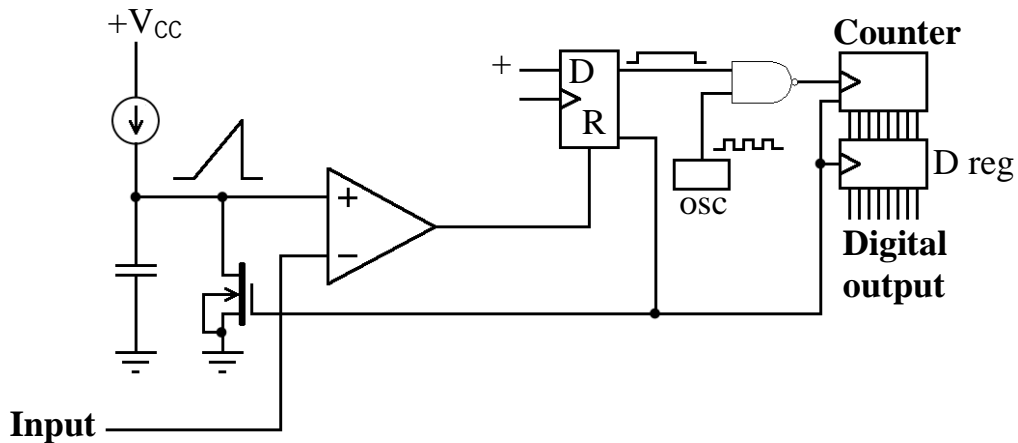
More gates can be added to this cascade as bits are needed for the register, but when more than two of these JK Flip-Flops are coupled together, sharing the same clock, *AND gates* must be used. To maintain the 2:1 gear functionality the information from the *all previous gates* have to be combined: Q_0 always toggles, Q_1 toggles if Q_0 is high, Q_2 toggles if Q_0 and Q_1 are high and Q_3 toggles if all the preceding outputs are high.



Why is counting important? - A few examples:

1. Counting is crucial in many nuclear and particle physics experiments. Here you count the **number of coincident particles** or the **number of photons** to hit a detector within a certain period of time. Instead of the circuit counting as driven by the clock, the *clock is replaced by a signal* generated by the detector electronics.
2. The **CCD chip** in your digital cameras relies on counting. Here no individual photons are counted in its pixels, instead the charge deposited by many incident photons is integrated over the exposure time. But when it comes to reading out the chip a clock signal starts which steers a circuit to push with each clock cycle the collected charges row-wise to the neighbouring pixels. (Abusing the above timing diagram: imagine the line Q_0 to represent one of the columns of the CCD, the 'wells' filled with charge. Now the structure moves sideways with each half clock cycle...). The pixel contents which come off the sensor at each clock cycle (e.g. at the right edge of the diagram) are collected in a single buffer line. That in turn is pushed with a high-speed clock sideways to a single channel charge-to-digital converter, digitising this way each pixel's content in turn. The bookkeeping of this process is done with ... you have three guesses..., with counters.

3. Finally, in the lab you are making use of a **multimeter** – located in the top right hand corner of the grey multi-function units. Counting is one of the basic operations of the **voltmeter**. A schematic is shown below. To measure you attach the voltmeter probe



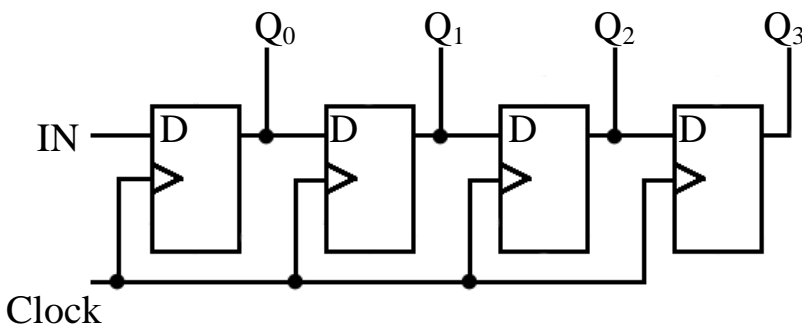
across some section of your circuit where you would like to know the difference in potential. You feed the 'high' line to the *input* of the circuit, the low line to the common *ground* (the upside down conifer-like symbols). An internal voltage within the voltmeter begins to ramp up from zero (the saw tooth shaped pulse). Together with this ramp a clock (osc) starts, feeding into the *counting circuit*. At each cycle the internal and external voltage are compared. At the clock cycle where the internal voltage passes the external the **comparator** (the triangle with the positive and the negative input) triggers the counter to stop. Looking up the calibration for the internal voltage ramp and for the range adjustment, the effective voltage is computed from the counting result.

Equipping the input stage with a calibrated resistance across which an input current produces a voltage drop, this circuit is converted to an **amperemeter**.

And by producing a calibrated voltage which then is sent through one of the resistors to produce a test current, the circuit is configured as an **ohmmeter** to measure the voltage drop across an external resistor. Use, reuse, recycle ...☺

D-Latch applications:

As you have seen earlier the **D-latch** is that configuration of the JK Flip-Flop the K and J inputs are forced to have opposite states. The operation table shows that type D-latches present at their output Q what was on their input J at the previous clock pulse. This is updated with every clock pulse. For such a simple operation it is surprising to find it has numerous uses, mainly in form of **shift registers**. Shift registers can be constructed from a series of D-latches as shown below. With each clock cycle the bit content of the register is shifted sideways by one position (the clock in the table below is running from top to bottom).

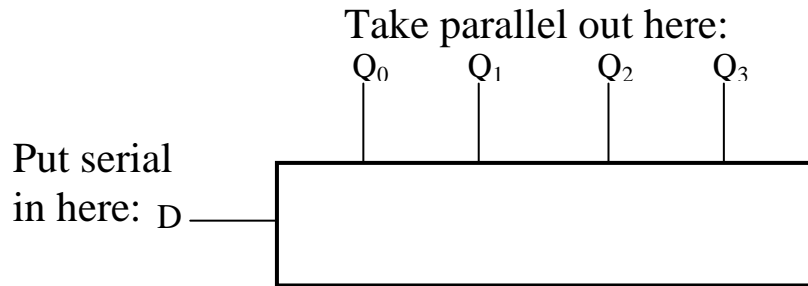


IN	Q ₀	Q ₁	Q ₂	Q ₃
1	0	0	0	0
0	1	0	0	0
0	0	1	0	0
0	0	0	1	0
0	0	0	0	1

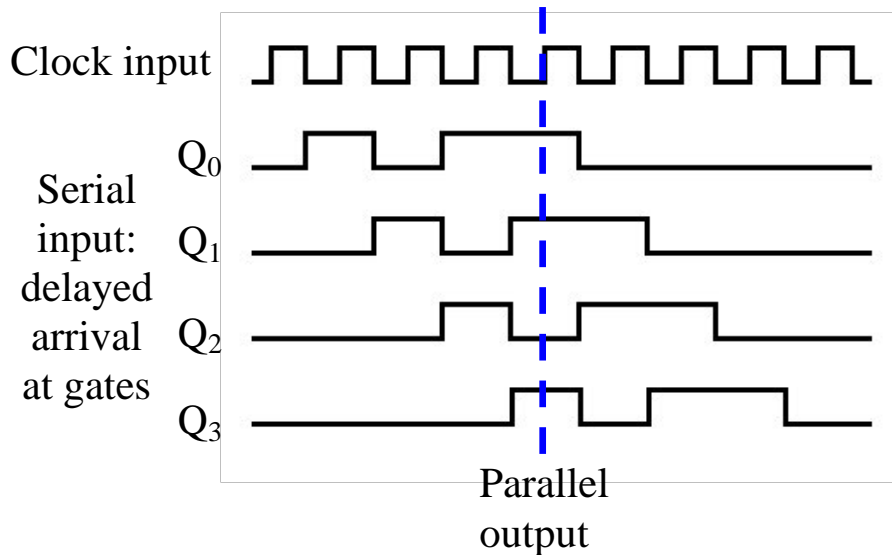
In a shift register which is holding an integer number the shift towards the most significant bit (Q_3 in the above table) **multiplies** its value **by two**, while a shift in the other direction **divides** its value **by two** (with an additional underflow gate catching the remainder).

Shift registers may be used as **digital delay lines** or as temporary memory units called **FIFO** memory (from first-in/first-out). These act like a pipe, with data put in at the one end and been pushed out again at the other (opposed to a **stack** which, as a first-in/last-out memory unit, rather behaves like a bucket).

Shift registers are also used for **serial to parallel conversion**, and vice versa. The representation of such a **de-serialiser** often looks similar to this:



What happens inside is that any complex serial signal that is put into the shift register (at D) makes its way across the latches synchronized with the clock pulses. The signal appears at each output Q_x with a delay, see the timing diagram below. At one clock cycle the serial signal is represented in parallel at all the output channels (indicated by the dashed line). That is the point to read out the parallelised signal.



Summary:

Synchronous circuits are controlled throughout by the **same clock signal**. By contrast *asynchronous* circuits have many different clocks and often have problems with delays.

Counting is used in applications from counting exotic particles to making a voltmeter. A robust counter can be built up from **JK Flip-Flops**.

The *shift register*, made from **D-latches**, is a work horse for digital signal processing.