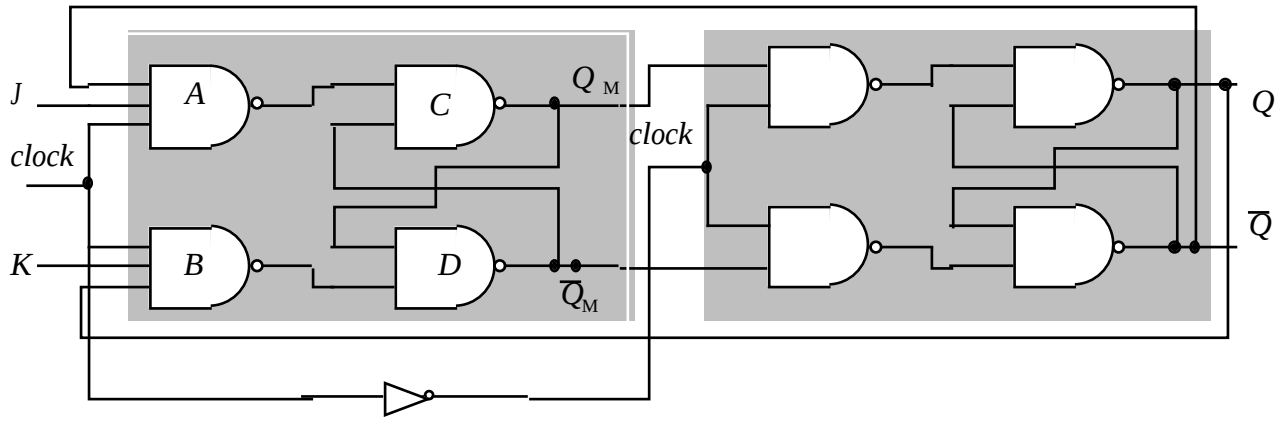
Week 5: More Sequential Logic

5.1.1 Master Slave JK Flip Flop

One might think that there would be few further variations of the flip flop possible, but there are still one or two, and we now come to an extremely common and useful circuit element: the *master-slave JK flip flop*. Its circuit is shown below. It can be analysed as two SR-like flip flops, but…

- The second flip flop is clocked with the complement of the clock input. Thus when the first flip flop is enabled (*clock*=1) the second is disabled, and then when the clock goes to zero the second flip flop is enabled and the first disabled.

The reason for this complication is to allow the *JK*=11 input condition to perform a useful function: that of "toggling" the outputs, so that $Q_{n+1} = \overline{Q}_n$ regardless of what value Q_n has. Between clock pulses $Q = Q_M$. When *JK* = 11 and the clock goes to 1, $\overline{Q}$ and Q fulfil the roles of *S* and *R* for the first flip flop – hence toggling its state because of the cross connection – while Q and $\overline{Q}$ themselves are held steady. At the end of the clock pulse the first flip flop becomes locked into its new state and that state is transferred to the second flip flop so that finally Q and $\overline{Q}$ change over.
Because Q and $\overline{Q}$ change only at the *trailing edge* of the clock pulse this is called an *edge-triggered* flip-flop, and since in this case it is at the 1→0 transition that the change occurs, this flip-flop is said to be *negative edge triggered* (assuming "1" is a more positive voltage than "0").

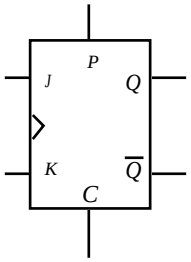


The truth table is

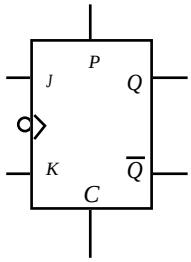
J	*K*	Q_{n+1}
0	0	Q_n
0	1	0
1	0	1
1	1	$\overline{Q}_n$

There are a few further subtleties. NAND gates *C* and *D* can be equipped with extra inputs that can be used to ***P**reset* the flip flop (Q =1), or ***C**lear* it (Q= 0) even when no clock pulse is applied.

Positive Going Edge Triggered Master Slave JK Flip Flop (Outputs change on positive going edge of clock)

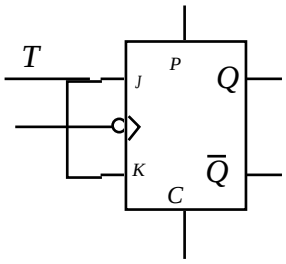


Negative Going Edge Triggered Master Slave JK Flip Flop (Outputs change on negative going edge of clock)



5.1.2 The T and D Flip Flops

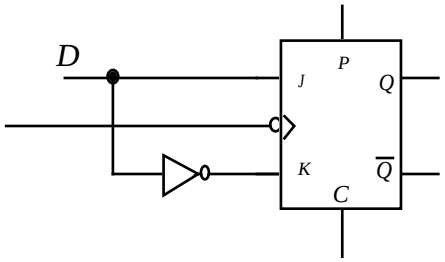
Finally we consider *T* and *D* flip flops which are specific configurations of *J K* flip flops.



A *T* flip flop is a *JK* flip flop connected with *J*=*K*="*T*", in which case the outputs always toggles (hence the "T") from one state to the other when *T*=1 (and remains unchanged when *T*=0).

Exercise 5.1 Confirm the operation of the *T* flip flop. Using *Crocodile Clips* confirm that the output toggles on the *positive edge* of the clock. [Note: *Crocodile Clips* uses positive edge triggered flip-flops whereas Beards tends to use negative edge triggering.]

A *D* flip flop is a *JK* flip flop connected with "$D" = J = \overline{K}$ in which case the output *Q* stores the value of *D* until the next clock pulse edge, even if *D* changes before then. It may also be constructed from an SR flip flop with $D = S = \overline{R}$. It therefore functions rather like a camera, taking a snap shot of *D* at the instant of the clock edge.



Exercise 5.2 Confirm the operation of the *D* flip flop. Confirm that the output stores the value of *D* at its output *Q* until the next clock pulse edge.

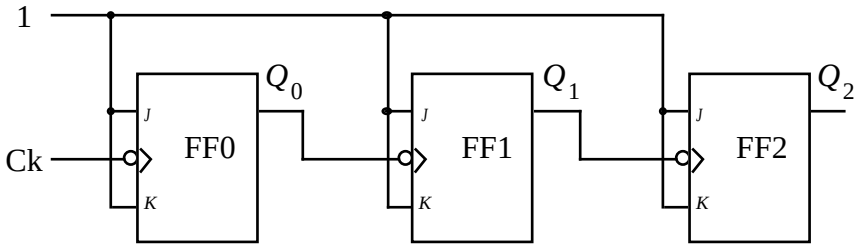
5.2 Asynchronous Counters

5.2.0 How can a circuit count?

Well intrinsically, it can't. But we can use electronic circuits to mimic what we do when we count, so if we use them correctly and interpret their output correctly, they perform the function of counting.

5.2.1 Ripple Counters

Ripple counters are the simplest form of counting circuit. Here is the "scale-of-8" ripple counter, so-called because it repeats its cycle every 8 clock pulses:



They are not usually the preferred choice for a counter because of

- frequency limitations.
- clumsiness (see 5.2.4 below)

Exercise 5.3 Confirm the operation of a scale-of-8 ripple counter. Using the positive edge triggered flip-flops in Crocodile Clips you should connect the $\overline{Q}$ outputs to the clock inputs, not the Q outputs as above. Can you see the circuit ripple on screen?

In exercises 5.3 to 5.5 you will need to take care over the type of triggering at the input to the JK flip flops. Beards assumes negative edge triggered flip flops, but this is not the most common type.

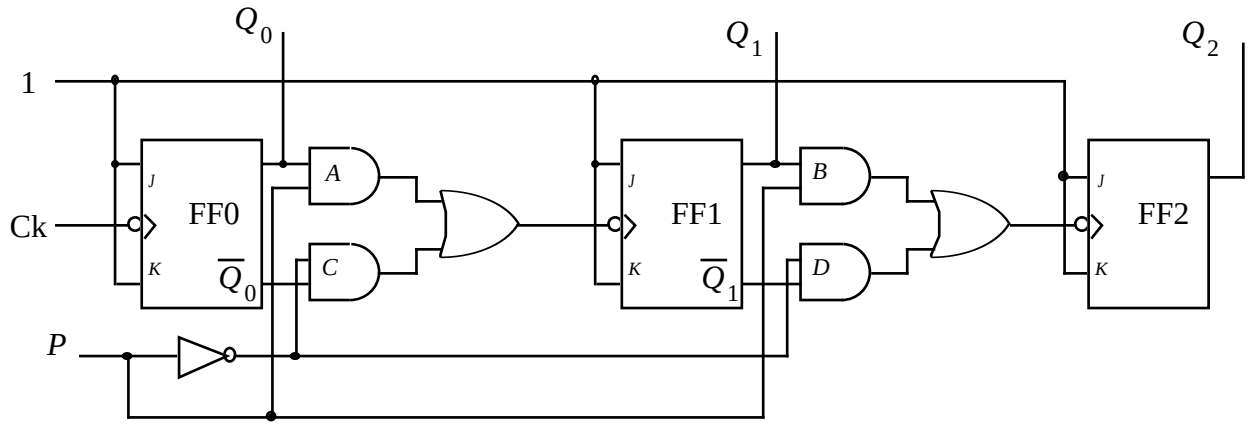
5.2.2 Count Down Ripple Counters

Notice that whether we have a count up or a count down circuit depends on whether Q or $\overline{Q}$ is connected to the clock input of the following flip flop and also on whether the flip flop has negative or positive edge triggering. Similarly, the $\overline{Q}$ outputs of an "up" counter will be counting down, and *vice versa*. There are therefore several ways to arrange for a counter to be counting up or down.

Exercise 5.4 Confirm the operation of the scale-of-8 count *down* ripple counter both by choosing the clock connections and by choosing the output connections.

5.2.3 Count Up *OR* Down Ripple Counters

Now we use inter-flip-flop logic to select one of either Q or $\overline{Q}$ to feed through to the next stage of the counter. Notice the use of AND logic to enable or disable inputs.

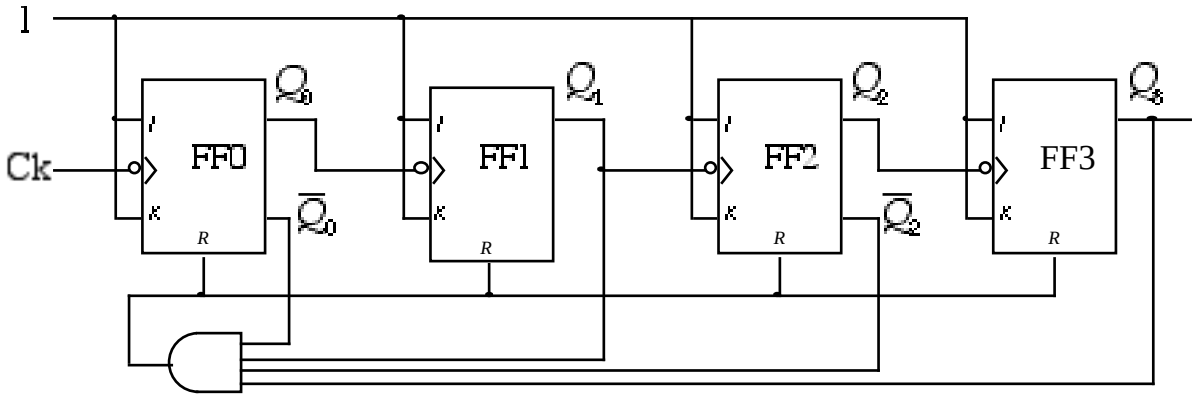


UP: $P = 1$ (or 0 if using positive edge triggering)
DOWN: $P = 0$ (or 1 if using positive edge triggering)

Exercise 5.5 Confirm the operation of a scale-of-8 count up *or* down ripple counter.

5.2.4 Decade Counters

Suppose we want a counter to reset after it has reached 9, i.e. we want it to count through a cycle of 10. We need to use four flip flops, but these would naturally count 0 to 15 (a cycle of 16). So we could use external logic to detect the appearance of ten (1010) and immediately cause all the flip flops to reset to zero. This technique could be used to reset the counters at any number less than the maximum count. The only problem is that… it doesn't work!



Exercise 5.6 Try it out in Crocodile Clips and try to understand what is happening. [The *clear* inputs in Crocodile Clips are labelled "R" and they are normal inputs not complementary ones as shown in Beards.] A challenge: can you make a *working* asynchronous decade counter? [Hint: you will need a further flip-flop or latch inserted into the *clear* signal after the AND gate to hold the clear signal on until the next clock.]

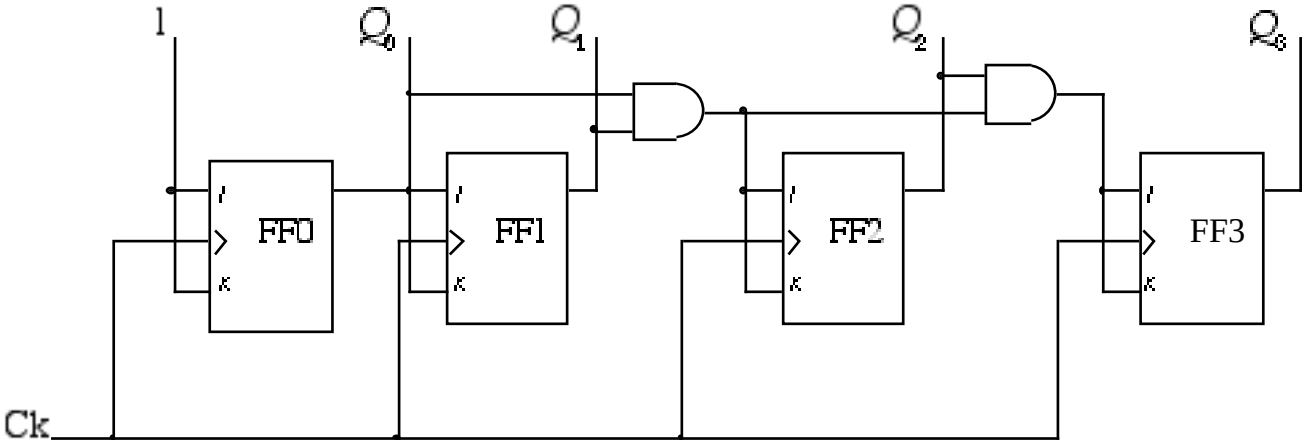
5.3 Synchronous Counters

5.3.0 Why bother?

We have seen that the action of a ripple counter "travels" through the flip flops. Thus before the circuit settles, it transiently shows a number of "wrong answers". Furthermore, every stage of counting takes a certain amount of time (≈ 20ns to 50ns), and so after say 8 stages of counting we must wait ≈ 0.5µs for the correct answer to appear. The circuit will not work at all if you try to make it count faster than this ripple time.

Synchronous counters yield the correct answer in one clock cycle and are correct after that time. The key to this technique is that all flip flops are clocked simultaneously and logic is used to determine whether or not to set the JK inputs of a particular stage to 1 so that the clock will cause that particular stage to toggle.

5.3.1 Scale of 16 Synchronous Counter



FF0 toggles every clock pulse. FF1 toggles only when Q_0 =1. FF2 toggles only when Q_0 =1 and Q_1 =1. FF3 toggles only when Q_0 =1 and Q_1 =1 and Q_2 =1.

Exercise 5.7 Confirm the operation of the scale-of-16 synchronous counter.

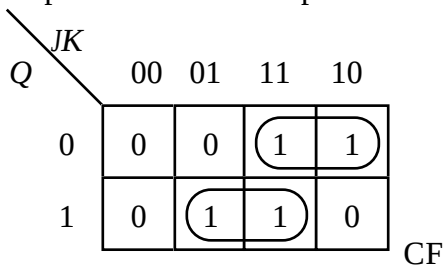
5.3.2 The "Change Function" Method

We need a general technique to derive the inter-stage logic for any synchronous counter. We can do this in a foolproof systematic way by using what is known as the *change function*. We first draw up a truth table which shows – for the type of flip flop we are using – every combination of *J*, *K* and *Q* and whether or not the flip flop should / would change when it is next clocked. We say the change function CF is 1 if the circuit toggles, other wise it is zero.

Change function for JK flip-flop:

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

The change function can then be plotted on a K-map as shown below:



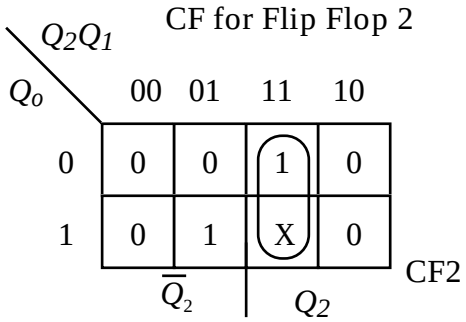
and we see that the CF for a JK flip flop is $CF = J\overline{Q} + KQ$

Now if we want to design, say (following Beards), a scale of 7 synchronous counter we see that there will be 3 flip flops involved (because $2^3 > 7$) and we must look at all the desired flip flop output states and ask: "Which flip flop must change (i.e. toggle) at the next clock pulse?"

We have

Q_2	Q_1	Q_0	$CF2$	$CF1$	$CF0$
0	0	0	0	0	1
0	0	1	0	1	1
0	1	0	0	0	1
0	1	1	1	1	1
1	0	0	0	0	1
1	0	1	0	1	1
1	1	0	1	1	0
0	0	0			

So if we plot the CF for say FF2 on a K-map we can deduce the appropriate logic:



(Note the "don't care" for the unused 111 condition)

Hence

$$\text{CF2} = Q_0 Q_1 \overline{Q}_2 + Q_1 Q_2$$

Note how we have deliberately arranged the K-map so that the central vertical line divides the map in two at the $Q/\overline{Q}$ boundary for the flip-flop which we are analysing and that we have avoided looping across that boundary, i.e. we have not produced a $Q_0 Q_1$ minterm.

As we deduced before, for any JK flip-flop its change function can always be written $CF = J\overline{Q} + KQ$ so by comparing these two equations we find expressions for J and K:

$$CF2 = Q_0 Q_1 \overline{Q}_2 + Q_1 Q_2$$

$$\equiv J_2 \overline{Q}_2 + K_2 Q_2$$

$$\therefore \quad J2 = Q_0 Q_1, \quad K2 = Q_1$$

This method usually produces the most efficient logic. An alternative and easier approach is to wire the flip-flop as a "T" flip-flop (J connected to K) and then fully minimise the CF K-map to give the logic to connect to T. In the current case this would mean including the second loop to give a change function of

$$CF2 = Q_0 Q_1 + Q_1 Q_2$$

$$= J_2 = K_2$$

Clearly this requires an extra AND gate (or an OR gate) compared to the first method. The best thing is to try both methods and choose the one that gives the minimum logic solution.

In a similar way, the logic for the other flip flops can be deduced. This allows us to systematically design synchronous counting logic.

- Work out the desired sequence of states;
- Work out the desired change functions;
- Minimise the CF K-maps to correspond to the JK form if possible;
- Work out the minimal logic for each FF in turn.

Exercise 5.8 Work out the CF for FF1 and FF0 and hence confirm that the circuit in Beards fig. 14.26 works as a scale of 7 synchronous counter.

5.4 Counter Applications

Exercise 5.9 Sketch out the design for a digital 24 hour "clock" (in the normal time-keeping sense) giving a BCD output of hours, minutes, seconds, tenths of seconds and hundredths of seconds, following the scheme outlined in Beards §14.7: the clock source is a 6.5536 MHz oscillator; this is followed by a divide-by-2^{16} counter to give a clock pulse of 100 Hz; then we have successive counters to divide by 10 to count the hundredths of seconds and produce a 10 Hz clock, divide by 10 to count the tenths of seconds and produce a 1 Hz clock, divide by 60 to count the seconds and produce a 1 per minute clock, divide by 60 to count the minutes and produce a 1 per hour clock, etc. How many flip flops would you need (roughly)?