Chapter 9

Digital Counter Design

A **Counter** is a device, which stores (and sometimes displays) the number of times a particular event has occurred, often in relationship to a CLOCK Signal. In electronics, **counters** can be implemented quite easily using memory devices such as **Flip-flops**. (Macao Museum of Communication , 2019)

Binary Counters

Thus we can see that a counter is nothing more than a specialized register or pattern generator that produces a specified output pattern or sequence of binary values (or states) upon the application of an input pulse signal called the "Clock".

The clock is actually used for data transfer in these applications. Typically, counters are logic circuits that can increment or decrement a count by one but when used as asynchronous divide-by-n counters they are able to divide these input pulses producing a clock division signal.

Counters are formed by connecting flip-flops together and any number of flip-flops can be connected or "cascaded" together to form a "divide-by-n" binary counter where "n" is the number of counter stages used and which is called the **Modulus**. The modulus or simply "MOD" of a counter is the number of output states the counter goes through before returning itself back to zero, one complete cycle.

Then a counter with three flip-flops like the circuit above will count from 0 to 7, 2^n-1. It has eight different output states representing the decimal numbers 0 to 7 and is called a **Modulo-8** or **MOD-8** counter. A counter with four flip-flops will count from 0 to 15 and is therefore called a **Modulo-16** counter and so on.

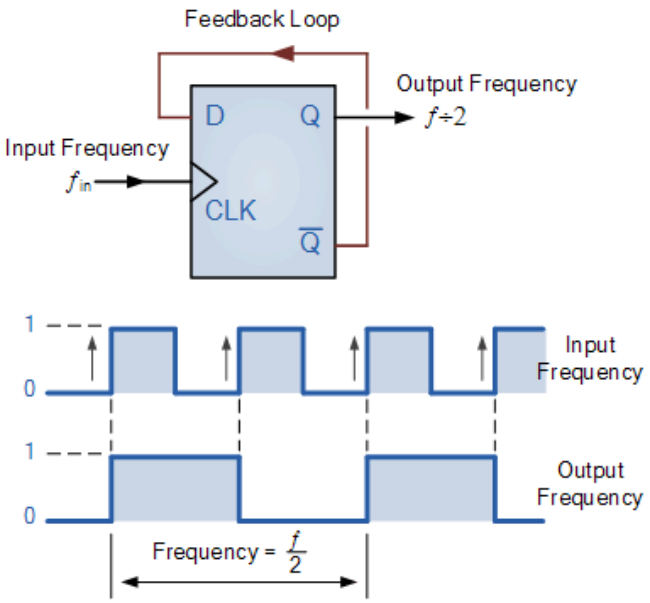
An example of this is given as.

- 3-bit Binary Counter = 2^3 = 8 (modulo-8 or MOD-8)
- 4-bit Binary Counter = 2^4 = 16 (modulo-16 or MOD-16)
- 8-bit Binary Counter = 2^8 = 256 (modulo-256 or MOD-256)

The Modulo number can be increased by adding more flip-flops to the counter and cascading is a method of achieving higher modulus counters. Then the modulo or MOD number can simply be written as: MOD number = 2^n

Divide-by-2 Counter

D-type Flip-Flop is also used as a binary divider, for **Frequency Division** or as a "divide-by-2" counter. Here the inverted output terminal Q (NOT-Q) is connected directly back to the Data input terminal D giving the device "feedback" as shown below.



It can be seen from the frequency waveforms above, that by "feeding back" the output from Q to the input terminal D, the output pulses at Q have a frequency that are exactly one half ($f \div 2$) that of the input clock frequency. In other words the circuit produces **Frequency Division** as it now divides the input frequency by a factor of two (an octave).

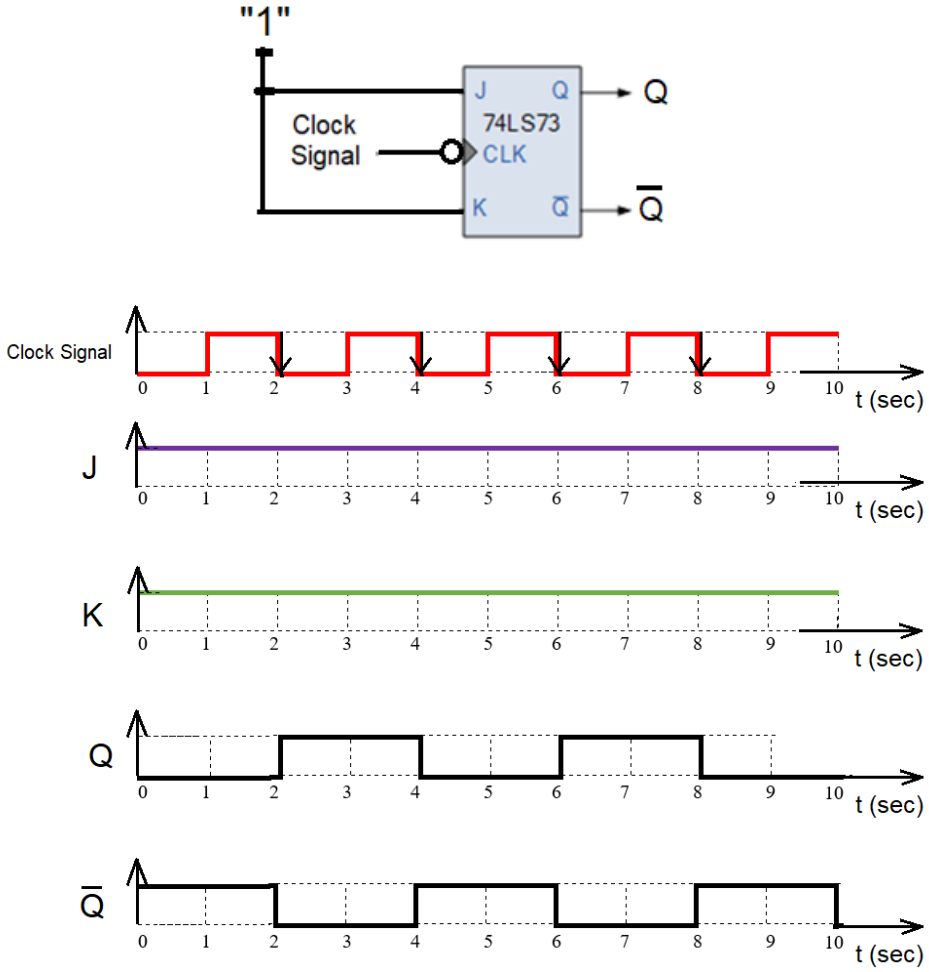
This then produces a type of counter called a "ripple counter" and in ripple counters, the clock pulse triggers the first flip-flop whose output triggers the second flip-flop, which in turn triggers the third flip-flop and so on through the chain producing a rippling effect (hence their name) of the timing signal as it passes through the chain.

The Toggle Flip-Flop

Another type of digital device that can be used for frequency division is the T-type or Toggle flip-flop. With a slight modification to a standard JK flip-flop, we can construct a new type of flip-flop called a **Toggle flip-flop**.

Toggle flip flops can be made from D-type flip-flops as shown above, or from standard JK flip-flops such as the 74LS73. The result is a device with only two inputs, the "Toggle" input itself and the negative controlling "Clock" input as shown.

74LS73 Toggle Flip Flop



A “Toggle flip-flop” gets its name from the fact that the flip-flop has the ability to toggle or switch between its two different states, the “toggle state” and the “memory state”. Since there are only two states, a T-type flip-flop is ideal for use in frequency division and binary counter design.

Binary ripple counters can be built using “Toggle” or “T-type flip-flops” by connecting the output of one to the clock input of the next. Toggle flip-flops are ideal for building ripple counters as it toggles from one state to the next, (HIGH to LOW or LOW to HIGH) at every clock cycle so simple frequency divider and ripple counter circuits can easily be constructed using standard T-type flip-flop circuits.

Asynchronous Counter

If we connect together in series, two T-type flip-flops the initial input frequency will be "divided-by-two" by the first flip-flop ($f \div 2$) and then "divided-by-two" again by the second flip-flop ($f \div 2$) $\div 2$, giving an output frequency which has effectively been divided four times, then its output frequency becomes one quarter value (25%) of the original clock frequency, ($f \div 4$).

Each time we add another toggle or "T-type" flip-flop to the chain, the output clock frequency is halved or divided-by-2 again and so on, giving an output frequency of 2^n where "n" is the number of flip-flops used in the sequence. Then the Toggle or T-type flip-flop is an edge triggered divide-by-2 device based upon the standard JK-type flip flop and which is triggered on the rising edge of the clock signal. The result is that each bit moves right by one flip-flop. All the flip-flops can be asynchronously reset and can be triggered to switch on either the leading or trailing edge of the input clock signal making it ideal for **Frequency Division**.

This type of counter circuit used for frequency division is commonly known as an **Asynchronous 3-bit Binary Counter** as the output on QA to QC, which is 3 bits wide, is a binary count from 0 to 7 for each clock pulse. Asynchronous Counters use flip-flops which are serially connected together so that the input clock pulse appears to ripple through the counter.

An **Asynchronous counter** can have 2^n-1 possible counting states MOD-16 for a 4-bit counter, (0-15) making it ideal for use in Frequency Division applications. But it is also possible to use the basic asynchronous counter configuration to construct special counters with counting states less than their maximum output number. For example, modulo or MOD counters.

This is achieved by forcing the counter to reset itself to zero at a pre-determined value producing a type of asynchronous counter that has truncated sequences. Then an n-bit counter that counts up to its maximum modulus (2^n) is called a full sequence counter and a n-bit counter whose modulus is less than the maximum possible is called a **truncated counter**.

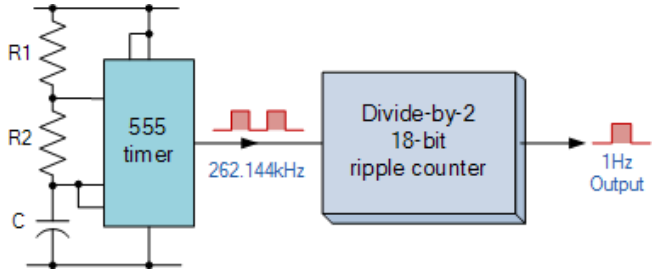
This ability of the ripple counter to truncate sequences to produce a "divide-by-n" output means that counters and especially ripple counters, can be used as frequency dividers to reduce a high clock frequency down to a more usable value for use in digital clocks and timing applications. For example, assume we require an accurate 1Hz timing signal to operate a digital clock.

We could quite easily produce a 1Hz square wave signal using a standard 555 timer chip configured as an Astable Multivibrator, but the manufacturers data sheet tells us that the 555

timer has a typical 1-2% timing error depending upon the manufacturer, and at low frequencies of 1Hz, this 2% timing error is not good.

However, the data sheet also tells us that the maximum operating frequency of the 555 timer is about 300kHz and a 2% error at this high frequency, while still large at about 6kHz maximum, would be acceptable. So by choosing a higher timing frequency of say 262.144kHz and an 18-bit ripple (Modulo-18) counter we can easily make a precision 1Hz timing signal as shown below.

1Hz timing signal from a 18-bit asynchronous ripple counter



This is of course a very simplistic example of how to produce accurate timing frequencies, but by using high frequency crystal oscillators and multi-bit frequency dividers, precision frequency generators can be produced for a full range of applications ranging from clocks or watches to event timing and even electronic piano/synthesizer or music type applications.

Unfortunately one of the main disadvantages with asynchronous counters is that there is a small delay between the arrival of the clock pulse at its input and it being present at its output due to the internal circuitry of the gate. In asynchronous circuits this delay is called the **Propagation Delay** giving the asynchronous ripple counter the nickname of "propagation counter" and in some high frequency cases this delay can produce false output counts.

In large bit ripple counter circuits, if the delay of the separate stages are all added together to give a summed delay at the end of the counter chain the difference in time between the input signal and the counted output signal can be very large. This is why the **Asynchronous Counter** is generally not used in high frequency counting circuits were large numbers of bits are involved.

Also, the outputs from the counter do not have a fixed time relationship with each other and do not occur at the same instant in time due to their clocking sequence. In other words the output frequencies become available one by one, a sort of domino effect. Then, the more flip-flops that are added to an asynchronous counter chain the lower the maximum operating frequency

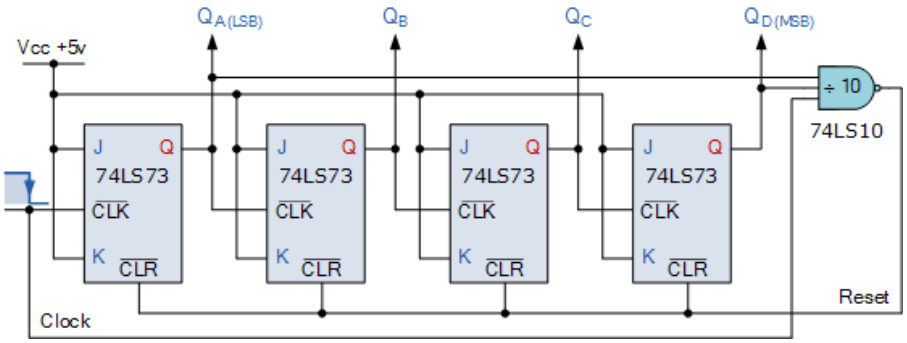
becomes to ensure accurate counting. To overcome the problem of propagation delay Synchronous Counters were developed.

Asynchronous Decade Counter

But why would we want to create an asynchronous truncated counter that is not a MOD-4, MOD-8, or some other modulus that is equal to the power of two. The answer is that we can by using combinational logic to take advantage of the asynchronous inputs on the flip-flop.

If we take the modulo-16 asynchronous counter and modified it with additional logic gates it can be made to give a decade (divide-by-10) counter output for use in standard decimal counting and arithmetic circuits.

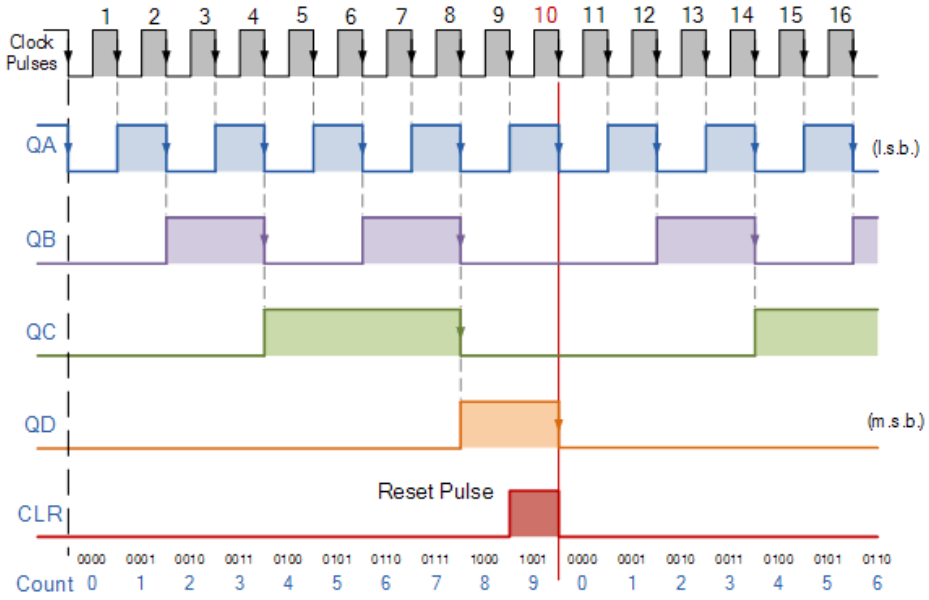
Such counters are generally referred to as **Decade Counters**. A decade counter requires resetting to zero when the output count reaches the decimal value of 10, ie. when DCBA = 1010 and to do this we need to feed this condition back to the reset input. A counter with a count sequence from binary "0000" (BCD = "0") through to "1001" (BCD = "9") is generally referred to as a BCD binary-coded-decimal counter because its ten state sequence is that of a BCD code but binary decade counters are more common.



This type of asynchronous counter counts upwards on each trailing edge of the input clock signal starting from 0000 until it reaches an output 1001 (decimal 9). Both outputs QA and QD are now equal to logic "1". On the application of the next clock pulse, the output from the 74LS10 NAND gate changes state from logic "1" to a logic "0" level.

As the output of the NAND gate is connected to the CLEAR (CLR) inputs of all the 74LS73 J-K Flip-flops, this signal causes all of the Q outputs to be reset back to binary 0000 on the count of 10. As outputs QA and QD are now both equal to logic "0" as the flip-flop's have just been reset, the output of the NAND gate returns back to a logic level "1" and the counter restarts again from 0000. We now have a decade or Modulo-10 up-counter.

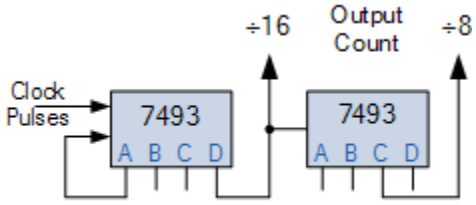
Decade Counter Timing Diagram



By using the same idea of truncating counter output sequences, the above circuit could easily be adapted to other counting cycles be simply changing the connections to the inputs of the NAND gate or by using other logic gate combinations. So for example, a scale-of-twelve (modulo-12) can easily be made by simply taking the inputs to the NAND gate from the outputs at "QC" and "QD", noting that the binary equivalent of 12 is 1100 and that output "QA" is the least significant bit (LSB).

Since the maximum modulus that can be implemented with n flip-flops is 2^n, this means that when you are designing truncated asynchronous counters you should determine the lowest power of two that is greater than or equal to your desired modulus. Let's say we wish to count from 0 to 39, or mod-40 and repeat. Then the highest number of flip-flops required would be six, n = 6 giving a maximum MOD of 64 as five flip-flops would not be enough as this only gives us a MOD-32.

Now suppose we wanted to build a "divide-by-128" counter for frequency division we would need to cascade seven flip-flops since $128 = 2^7$. Using dual flip-flops such as the 74LS74 we would still need four IC's to complete the circuit.



Divide-by-128 circuit

One easy alternative method would be to use two TTL 7493's as 4-bit ripple counter/dividers. Since 128 = 16 x 8, one 7493 could be configured as a "divide-by-16" counter and the other as a "divide-by-8" counter. The two IC's would be cascaded together to form a "divide-by-128" frequency divider as shown.

Of course standard IC asynchronous counters are available such as the TTL 74LS90 programmable ripple counter/divider which can be configured as a divide-by-2, divide-by-5 or any combination of both. The 74LS390 is a very flexible dual decade driver IC with a large number of "divide-by" combinations available ranging form divide-by-2, 4, 5, 10, 20, 25, 50, and 100.

In an asynchronous counter, the clock is applied only to the first stage with the output of one flip-flop stage providing the clocking signal for the next flip-flop stage and subsequent stages derive the clock from the previous stage with the clock pulse being halved by each stage.

This arrangement is commonly known as *Asynchronous* as each clocking event occurs independently as all the bits in the counter do not all change at the same time. As the counter counts sequentially in an upwards direction from 0 to 7. This type of counter is also known as an "up" or "forward" counter (**CTU**) or a **"3-bit Asynchronous Up Counter"**. The three-bit asynchronous counter shown is typical and uses flip-flops in the toggle mode. Asynchronous "Down" counters (**CTD**) are also available.

Therefore we can see that the output from the D-type flip-flop is at half the frequency of the input, in other words it counts in 2's. By cascading together more D-type or Toggle Flip-Flops, we can produce a divide-by-2, divide-by-4, divide-by-8, etc. circuit which will divide the input clock frequency by 2, 4 or 8 times, in fact any value to the power-of-2 we want making a binary counter circuit.

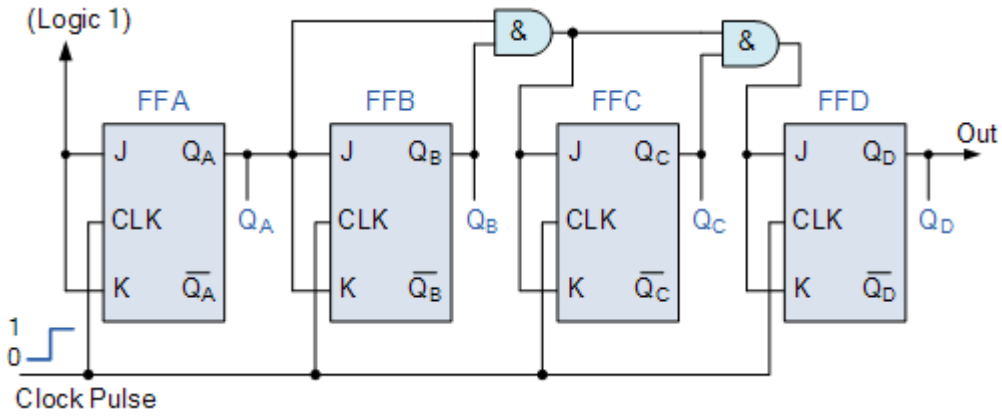
Multi-bit asynchronous counters connected in this manner are also called **"Ripple Counters"** or ripple dividers because the change of state at each stage appears to "ripple" itself through the counter from the LSB output to its MSB output connection. Ripple counters are available in standard IC form, from the 74LS393 Dual 4-bit counter to the 74HC4060, which is a 14-bit ripple counter with its own built in clock oscillator and produce excellent frequency division of the fundamental frequency.

Synchronous Counter

Synchronous Counters are so called because the clock input of all the individual flip-flops within the counter are all clocked together at the same time by the same clock signal. The result of this is that the Asynchronous counter suffers from what is known as “Propagation Delay” in which the timing signal is delayed a fraction through each flip-flop.

However, with the **Synchronous Counter**, the external clock signal is connected to the clock input of EVERY individual flip-flop within the counter so that all of the flip-flops are clocked together simultaneously (in parallel) at the same time giving a fixed time relationship. In other words, changes in the output occur in “synchronization” with the clock signal. The result of this synchronization is that all the individual output bits changing state at exactly the same time in response to the common clock signal with no ripple effect and therefore, no propagation delay.

Binary 4-bit Synchronous Up Counter

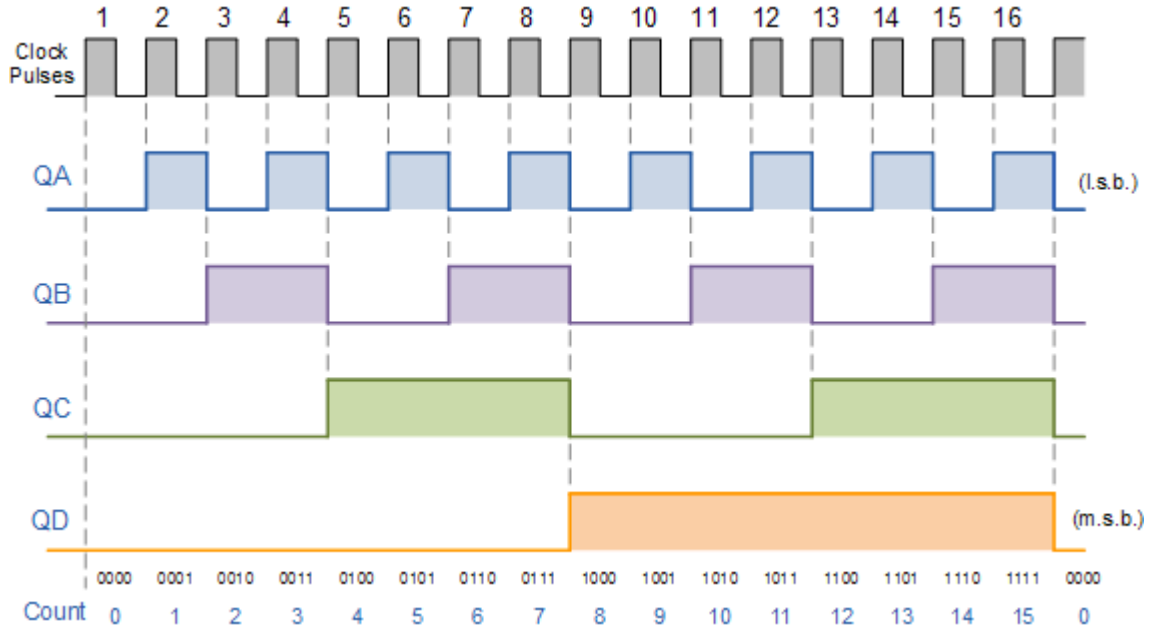


It can be seen above, that the external clock pulses (pulses to be counted) are fed directly to each of the **J-K flip-flops** in the counter chain and that both the J and K inputs are all tied together in toggle mode, but only in the first flip-flop, flip-flop FFA (LSB) are they connected HIGH, logic “1” allowing the flip-flop to toggle on every clock pulse. Then the synchronous counter follows a predetermined sequence of states in response to the common clock signal, advancing one state for each pulse.

The J and K inputs of flip-flop FFB are connected directly to the output Q_A of flip-flop FFA, but the J and K inputs of flip-flops FFC and FFD are driven from separate AND gates which are also supplied with signals from the input and output of the previous stage. These additional AND gates generate the required logic for the JK inputs of the next stage.

If we enable each JK flip-flop to toggle based on whether or not all preceding flip-flop outputs (Q) are “HIGH” we can obtain the same counting sequence as with the asynchronous circuit but without the ripple effect, since each flip-flop in this circuit will be clocked at exactly the same time. Then as there is no inherent propagation delay in synchronous counters, because all the counter stages are triggered in parallel at the same time, the maximum operating frequency of this type of frequency counter is much higher than that for a similar asynchronous counter circuit.

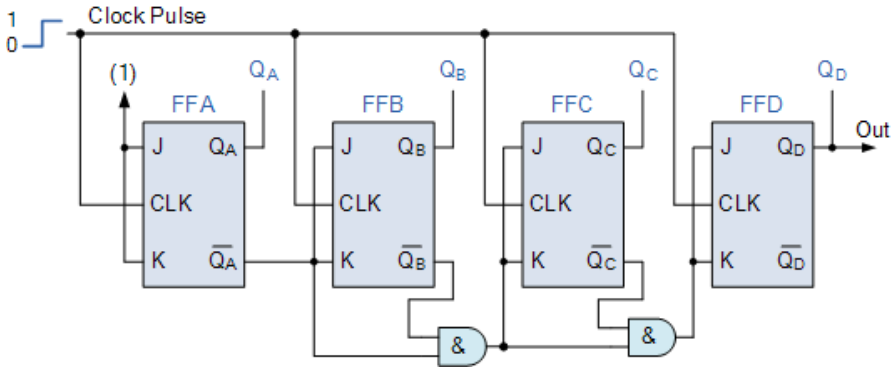
4-bit Synchronous Counter Waveform Timing Diagram



Because this 4-bit synchronous counter counts sequentially on every clock pulse the resulting outputs count upwards from 0 (0000) to 15 (1111). Therefore, this type of counter is also known as a **4-bit Synchronous Up Counter**.

However, we can easily construct a **4-bit Synchronous Down Counter** by connecting the AND gates to the Q output of the flip-flops as shown to produce a waveform timing diagram the reverse of the above. Here the counter starts with all of its outputs HIGH (1111) and it counts down on the application of each clock pulse to zero, (0000) before repeating again.

Binary 4-bit Synchronous Down Counter

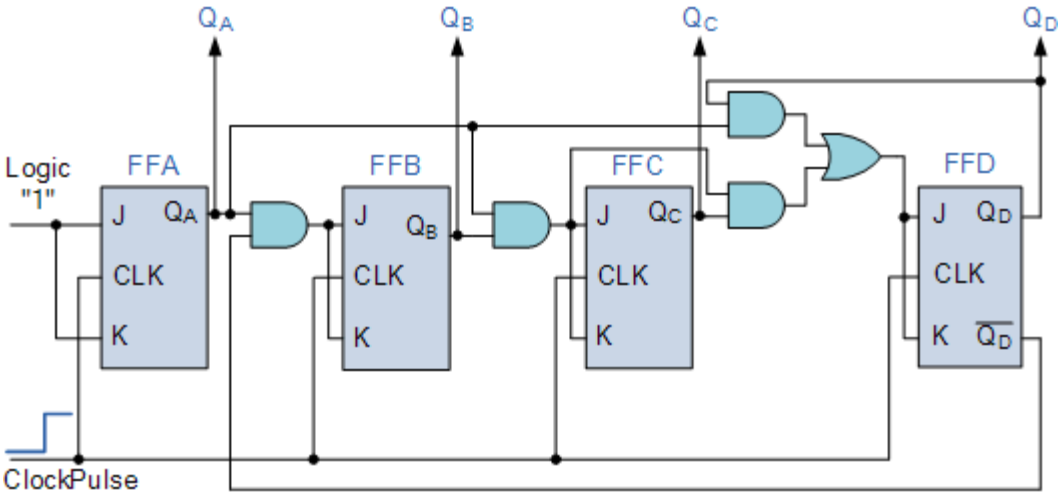


As synchronous counters are formed by connecting flip-flops together and any number of flip-flops can be connected or "cascaded" together to form a "divide-by-n" binary counter, the modulo's or "MOD" number still applies as it does for asynchronous counters so a Decade counter or BCD counter with counts from 0 to 2^n-1 can be built along with truncated sequences. All we need to increase the MOD count of an up or down synchronous counter is an additional flip-flop and AND gate across it.

Decade 4-bit Synchronous Counter

A 4-bit decade synchronous counter can also be built using synchronous binary counters to produce a count sequence from 0 to 9. A standard binary counter can be converted to a decade (decimal 10) counter with the aid of some additional logic to implement the desired state sequence. After reaching the count of "1001", the counter recycles back to "0000". We now have a decade or **Modulo-10** counter.

Decade 4-bit Synchronous Counter



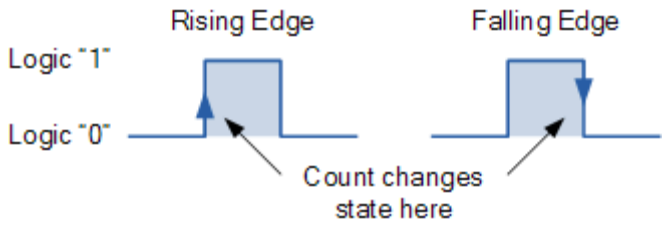
The additional AND gates detect when the counting sequence reaches “1001”, (Binary 10) and causes flip-flop FF3 to toggle on the next clock pulse. Flip-flop FF0 toggles on every clock pulse. Thus, the count is reset and starts over again at “0000” producing a synchronous decade counter.

We could quite easily re-arrange the additional AND gates in the above counter circuit to produce other count numbers such as a Mod-12 counter which counts 12 states from”0000″ to “1011” (0 to 11) and then repeats making them suitable for clocks, etc.

Triggering A Synchronous Counter

Synchronous Counters use edge-triggered flip-flops that change states on either the “positive-edge” (rising edge) or the “negative-edge” (falling edge) of the clock pulse on the control input resulting in one single count when the clock input changes state.

Generally, synchronous counters count on the rising-edge which is the low to high transition of the clock signal and asynchronous ripple counters count on the falling-edge which is the high to low transition of the clock signal.



It may seem unusual that ripple counters use the falling-edge of the clock cycle to change state, but this makes it easier to link counters together because the most significant bit (MSB) of one counter can drive the clock input of the next.

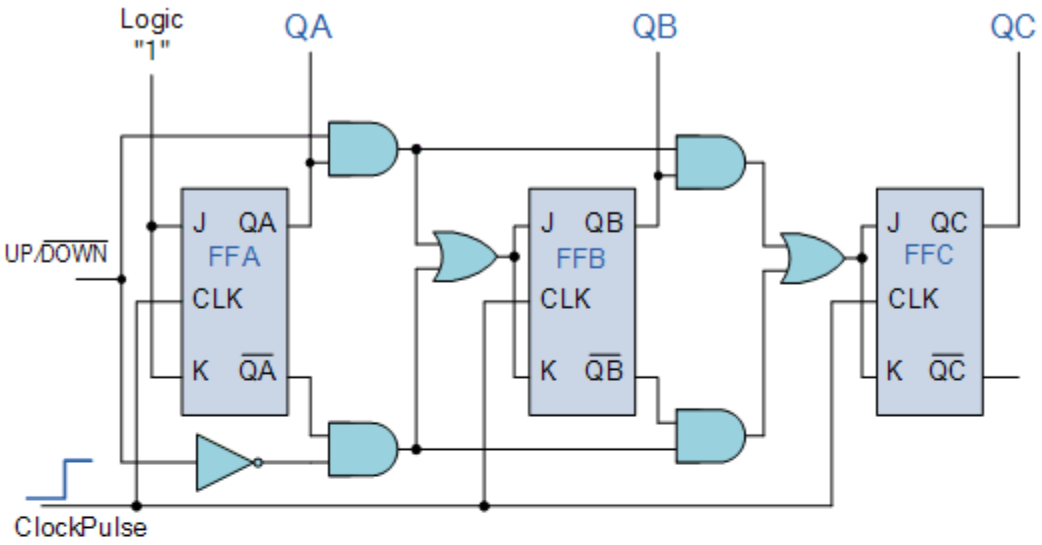
This works because the next bit must change state when the previous bit changes from high to low – the point at which a carry must occur to the next bit. Synchronous counters usually have a carry-out and a carry-in pin for linking counters together without introducing any propagation delays.

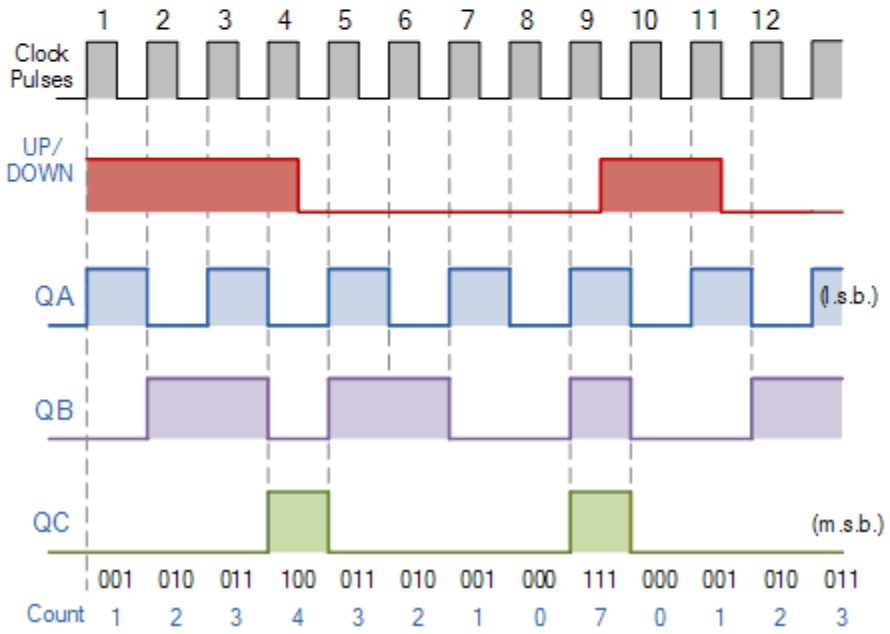
Bidirectional Counters

Both Synchronous and Asynchronous counters are capable of counting “Up” or counting “Down”, but their is another more “Universal” type of counter that can count in both directions either Up or Down depending on the state of their input control pin and these are known as **Bidirectional Counters**.

Bidirectional counters, also known as Up/Down counters, are capable of counting in either direction through any given count sequence and they can be reversed at any point within their count sequence by using an additional control input as shown below.

Synchronous 3-bit Up/Down Counter





The circuit above is of a simple 3-bit Up/Down synchronous counter using JK flip-flops configured to operate as toggle or T-type flip-flops giving a maximum count of zero (000) to seven (111) and back to zero again. Then the 3-Bit counter advances upward in sequence (0,1,2,3,4,5,6,7) or downwards in reverse sequence (7,6,5,4,3,2,1,0).

Generally most bidirectional counter chips can be made to change their count direction either up or down at any point within their counting sequence. This is achieved by using an additional input pin which determines the direction of the count, either Up or Down and the timing diagram gives an example of the counters operation as this Up/Down input changes state.

Nowadays, both up and down counters are incorporated into single IC that is fully programmable to count in both an "Up" and a "Down" direction from any preset value producing a complete **Bidirectional Counter** chip. Common chips available are the 74HC190 4-bit BCD decade Up/Down counter, the 74F569 is a fully synchronous Up/Down binary counter and the CMOS 4029 4-bit Synchronous Up/Down counter.

MOD Counters

MOD Counters are cascaded counter circuits which count to a set modulus value before resetting. The job of a counter is to count by advancing the contents of the counter by one count with each clock pulse. Counters which advance their sequence of numbers or states when activated by a clock input are said to operate in a "count-up" mode. Likewise, counters which decrease their sequence of numbers or states when activated by a clock input are said to operate in a "count-down" mode. Counters that operate in both the UP and DOWN modes, are called bidirectional counters.

Counters are sequential logic devices that are activated or triggered by an external timing pulse or clock signal. A counter can be constructed to operate as a synchronous circuit or as an asynchronous circuit. With synchronous counters, all the data bits change synchronously with the application of a clock signal. Whereas an asynchronous counter circuit is independent of the input clock so the data bits change state at different times one after the other.

Then counters are sequential logic devices that follow a predetermined sequence of counting states which are triggered by an external clock (CLK) signal. The number of states or counting sequences through which a particular counter advances before returning once again back to its original first state is called the **modulus** (MOD). In other words, the modulus (or just modulo) is the number of states the counter counts and is the dividing number of the counter.

Modulus Counters, or simply *MOD counters*, are defined based on the number of states that the counter will sequence through before returning back to its original value. For example, a 2-bit counter that counts from 00_2 to 11_2 in binary, that is 0 to 3 in decimal, has a modulus value of 4 ($00 \rightarrow 1 \rightarrow 10 \rightarrow 11$, and return back to 00) so would therefore be called a modulo-4, or mod-4, counter. Note also that it has taken four clock pulses to get from 00 to 11. As in this simple example there are only two bits, ($n = 2$) then the maximum number of possible output states (maximum modulus) for the counter is: $2^n = 2^2$ or 4. However, counters can be designed to count to any number of 2^n states in their sequence by cascading together multiple counting stages to produce a single modulus or MOD-N counter.

Therefore, a "Mod-N" counter will require "N" number of flip-flops connected together to count a single data bit while providing 2^n different output states, (n is the number of bits). Note that N is always a whole integer value.

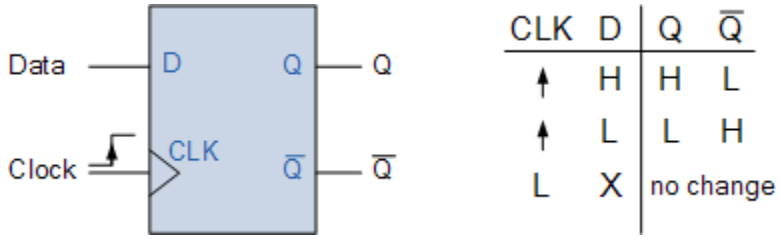
The we can see that MOD counters have a modulus value that is an integral power of 2, that is, 2, 4, 8, 16 and so on to produce an n-bit counter depending on the number of flip-flops used, and how they are connected, determining the type and modulus of the counter.

Digital Counter Design using D flip-flops

MOD counters are made using "flip-flops" and a single flip-flop can produce a count of 0 or 1, giving a maximum count of 2. There are different types of flip-flop designs we could use, the S-R, the J-K, J-K Master-slave, the D-type or even the T-type flip-flop to construct a counter. But to keep things simple, we will use the D-type flip-flop, (DFF) also known as a Data Latch, because a single data input and external clock signal are used, and is also positive edge triggered.

The D-type flip-flop, such as the TTL 74LS74, can be made from either S-R or J-K based edge-triggered flip-flops depending on whether you want it to change state either on the positive or leading edge (0 to 1 transition) or on the negative or trailing edge (1 to 0 transition) of the clock pulse. Here we will assume a positive, leading-edge triggered flip-flop. You can find more information in the following link about **D-type flip-flops**.

D-type Flip-flop and Truth Table



CLK	D	Q	$\overline{Q}$
↑	H	H	L
↑	L	L	H
L	X	no change	

The operation of a D-type flip-flop, (DFF) is very simple as it only has a single data input, called "D", and an additional clock "CLK" input. This allows a single data bit (0 or 1) to be stored under the control of the clock signal thus making the D-type flip-flop a synchronous device because the data on the inputs is transferred to the flip-flops output only on the triggering edge of the clock pulse.

So from the truth table, if there is a logic "1" (HIGH) on the Data input when a positive clock pulse is applied, the flip-flop SET's and stores a logic "1" at "Q", and a complimentary "0" at Q. Likewise, if there is a LOW on the Data input when another positive clock pulse is applied, the flip-flop RESET's and stores a "0" at "Q", and a resulting "1" at Q. Then the output "Q" of the D-type flip-flop responds to the value of the input "D" when the clock (CLK) input is HIGH. When the clock input is LOW, the condition at "Q", either "1" or "0" is held until the next time

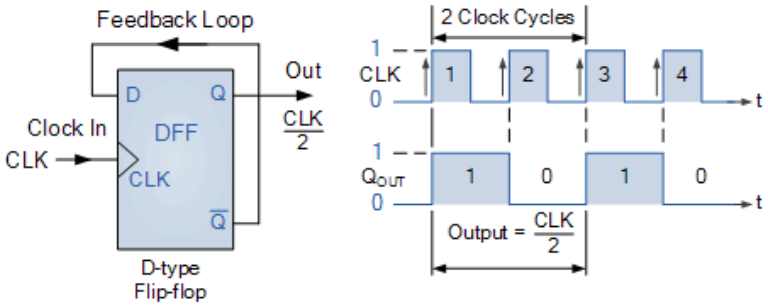
the clock signal goes HIGH to logic level "1". Therefore the output at "Q" only changes state when the clock input changes from a "0" (LOW) value to a "1" (HIGH) making it a positive edge triggered D-type flip-flop. Note that negative edge-triggered flip-flops work in exactly the same way except that the falling edge of the clock pulse is the triggering edge.

So now we know how an edge-triggered D-type flip-flop works, lets look at connecting some together to form a MOD counter.

Divide-by-Two Counter

The edge-triggered D-type flip-flop is a useful and versatile building block to construct a MOD counter or any other type of sequential logic circuit. By connecting the Q output back to the "D" input as shown, and creating a feedback loop, we can convert it into a binary divide-by-two counter using the clock input only as the Q output signal is always the inverse of the Q output signal.

Divide-by-two Counter and Timing Diagram



The timing diagrams show that the "Q" output waveform has a frequency exactly one-half that of the clock input, thus the flip-flop acts as a frequency divider. If we added another D-type flip-flop so that the output at "Q" was the input to the second DFF, then the output signal from this second DFF would be one-quarter of the clock input frequency, and so on. So for an "n" number of flip-flops, the output frequency is divided by 2n, in steps of 2.

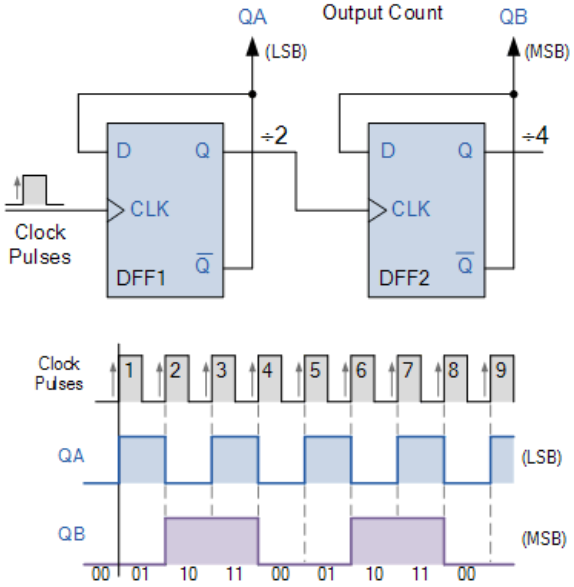
Note that this method of **frequency division** is very handy for use in sequential counting circuits. For example, a 60Hz mains frequency signal could be reduced to a 1Hz timing signal by using a divide-by-60 counter. A divide-by-6 counter would divide the 60Hz down to 10Hz which is then feed to a divide-by-10 counter to divide the 10Hz down to a 1Hz timing signal or pulse, etc.

MOD-4 Counter

Technically as well as being a 1-bit storage device, a single flip-flop on its own could be thought of as a MOD-2 counter, as it has a single output resulting in a count of two, either a 0 or 1, on the application of the clock signal. But a single flip-flop on its own produces a limited counting sequence, so by connecting together more flip-flops to form a chain, we can increase the counting capacity and construct a MOD counter of any value.

If a single flip-flop can be considered as a modulo-2 or MOD-2 counter, then adding a second flip-flop would give us a MOD-4 counter allowing it to count in four discrete steps. The overall effect would be to divide the original clock input signal by four. Then the binary sequence for this 2-bit MOD-4 counter would be: 00, 01, 10, and 11 as shown.

MOD-4 Counter and Timing Diagram

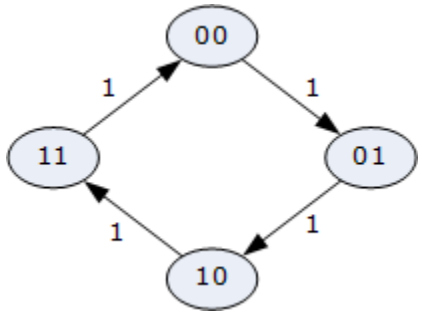


Note that for simplicity, the switching transitions of QA, QB and CLK in the above timing diagram are shown to be simultaneous even though this connection represents an asynchronous counter. In reality there would be a very small switching delay between the application of the positive going clock (CLK) signal, and the outputs at QA and QB.

We can show visually the operation of this 2-bit asynchronous counter using a truth table and state diagram.

MOD-4 Counter State Diagram

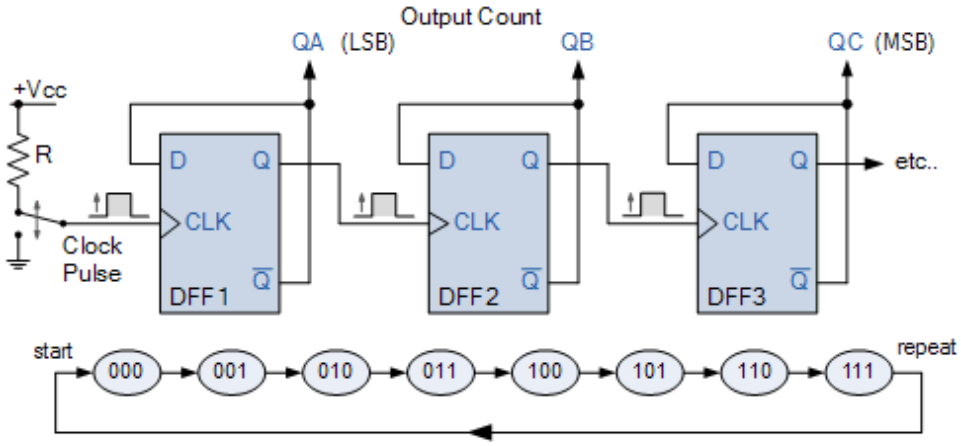
Clock Pulse	Present State			Next State		State Diagram
	Q_B	Q_A		Q_B	Q_A	
0 (start)	0	0	⇒	0	1	
1	0	1	⇒	1	0	
2	1	0	⇒	1	1	
3	1	1	⇒	0	0	
4 (repeat)	0	0	⇒	0	1	



We can see from the truth table of the counter, and by reading the values of QA and QB, when QA = 0 and QB = 0, the count is 00. After the application of the clock pulse, the values become QA = 1, QB = 0, giving a count of 01 and after the next clock pulse, the values become QA = 0, QB = 1, giving a count of 10. Finally the values become QA = 1, QB = 1, giving a count of 11. The application of the next clock pulse causes the count to return back to 00, and thereafter it counts continuously up in a binary sequence of: 00, 01, 10, 11, 00, 01 …etc.

Then we have seen that a MOD-2 counter consists of a single flip-flop and a MOD-4 counter requires two flip-flops,allowing it to count in four discrete steps. We could easily add another flip-flop onto the end of a MOD-4 counter to produce a MOD-8 counter giving us a 2^3 binary sequence of counting from 000 up to 111, before resetting back to 000. A fourth flip-flop would make a MOD-16 counter and so on, in fact we could go on adding extra flip-flops for as long as we wanted.

MOD-8 Counter and State Diagram



State transition diagram for a MOD-8 binary counter

We can therefore construct mod counters to have a natural count of 2^n states giving counters with mod counts of 2, 4, 8, 16, and so on, before repeating itself. But sometimes it is necessary to have a modulus counter that resets its count back to zero during the normal counting process and does not have a modulo that is a power of 2. For example, a counter having a modulus of 3, 5, 6, or 10.

Counters of Modulo "m"

Counters, either synchronous or asynchronous progress one count at a time in a set binary progression and as a result an "n"-bit counter functions naturally as a modulo 2^n counter. But we can construct mod counters to count to any value we want by using one or more external logic gates causing it to skip a few output states and terminate at any count resetting the counter back to zero, that is all flip-flops have Q = 0.

In the case of modulo "m" counters, they do not count to all their possible states, but instead count to the "m" value and then return to zero. Obviously, "m" is a number smaller than 2^n, (**$m < 2^n$**). So how do we get a binary counter to return to zero part way through its count.

Fortunately, as well as counting, up or down, counters can also have additional inputs called *CLEAR* and *PRESET* which makes it possible to clear the count to zero, (all Q = 0) or to preset the counter to some initial value. The TTL 74LS74 has active-low Preset and Clear inputs.

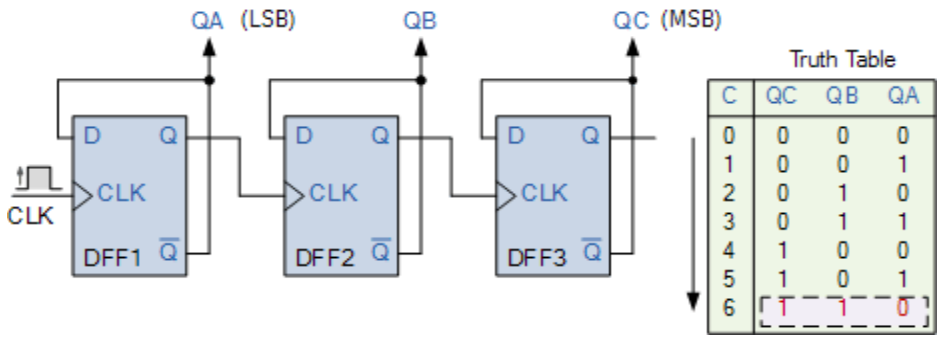
Let's assume for simplicity that the CLEAR inputs are all connected together and are active-high inputs allowing the flip-flops to operate normally when the Clear input is equal to 0 (LOW). But if the Clear input is at logic level "1" (HIGH), then the next positive edge of the clock signal will reset all the flip-flops into the state Q = 0, regardless of the value of the next clock signal.

Note also that as all the Clear inputs are connected together, a single pulse can also be used to clear the outputs (Q) of all the flip-flops to zero before counting starts to ensure that the count actually starts from zero. Also some larger bit counters have an additional ENABLE or INHIBIT input pin which allows the counter to stop the count at any point in the counting cycle and hold its present state, before being allowed to continue counting again. This means the counter can be stopped and started at will without resetting the outputs to zero.

A Modulo-5 Counter

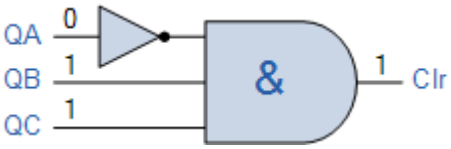
Suppose we want to design a MOD-5 counter, how could we do that. First we know that "m = 5", so 2^n must be greater than 5. As $2^1 = 2$, $2^2 = 4$, $2^3 = 8$, and 8 is greater than 5, then we need a counter with three flip-flops (N = 3) giving us a natural count of 000 to 111 in binary (0 to 7 decimal).

Taking the MOD-8 counter above, the truth table for the natural count is given as:



As we are constructing a MOD-5 counter, we want the counter to reset back to zero after a count of 5. However, we can see from the attached truth table that the count of six gives us the output condition of: QA = 0, QB = 1, and QC = 1.

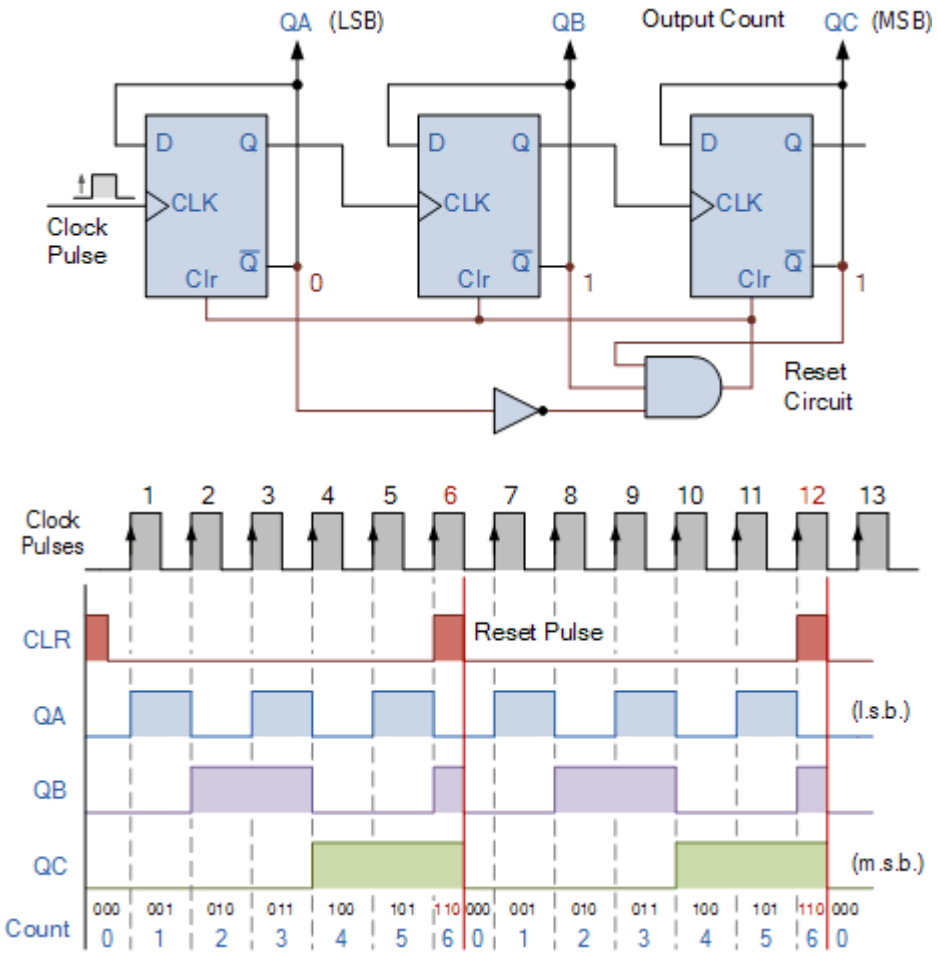
We can decode this output state of 011 (6) to give us a signal to clear (Clr) the counter back to zero with the help of a 3-input AND gate (TTL 74LS11) and an inverter or NOT gate, (TTL 74LS04).



The inputs of the combinational logic circuit of the inverter and AND gate are connected to QA, QB, and QC respectively with the output of the AND gate at logic level “0” (LOW) for any combinations of the input other than the one we want.

In binary code, the output sequence count will look like this: 000, 001, 010, 011, 100, 101. But when it reaches the state of 011 (6), the combinational logic circuit will detect this 011 state and produce an output at logic level “1” (HIGH).

We can then use the resulting HIGH output from the AND gate to reset the counter back to zero after its output of 5 (decimal) count giving us the required MOD-5 counter. When the output from the combinational circuit is LOW it has no effect on the counting sequence.



Then we can use combinational logic decoding circuits around a basic counter, either synchronous or asynchronous to produce any type of MOD Counter we require as each of the counters unique output states can be decoded to reset the counter at the desired count.

In our simple example above, we have used a 3-input AND gate to decode the 011 state, but the first time that QA and QB are both at logic 1 is is when the count reaches six, so a 2-input AND gate connected to QA and QB could be used without the complication of the third input and the inverter.

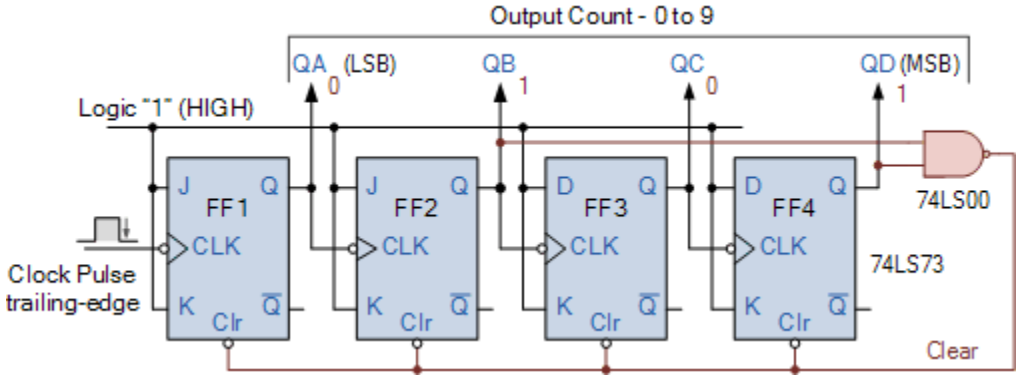
However, one of the disadvantages of using asynchronous counters for producing a MOD counter of a desired count is that undesired effects called “glitches” can occur when the counter reaches its reset condition. During this brief time the outputs of the counter may take on an incorrect value, so it is sometimes better to use synchronous counters as modulo-m counters as all the flip-flops are clocked by the same clock signal so change state at the same time.

Modulus 10 Counter

A good example of a modulo-m counter circuit which uses external combinational circuits to produce a counter with a modulus of 10 is the Decade Counter. Decade (divide-by-10) counters such as the TTL 74LS90, have 10 states in its counting sequence making it suitable for human interfacing where a digital display is required.

The decade counter has four outputs producing a 4-bit binary number and by using external AND and OR gates we can detect the occurrence of the 9th counting state to reset the counter back to zero. As with other mod counters, it receives an input clock pulse, one by one, and counts up from 0 to 9 repeatedly.

Once it reaches the count 9 (1001 in binary), the counter goes back to 0000 instead of continuing on to 1010. The basic circuit of a decade counter can be made from JK flip-flops (TTL 74LS73) that switch state on the negative trailing-edge of the clock signal as shown.



MOD Counter Summary

We have seen in this tutorial about **MOD Counters** that binary counters are sequential circuits that generate binary sequences of bits as a result of a clock signal and the state of a binary counter is determined by the specific combination formed by all the counters outputs together.

The number of different output states a counter can produce is called the modulo or modulus of the counter. The Modulus (or MOD-number) of a counter is the total number of unique states it passes through in one complete counting cycle with a mod-n counter being described also as a divide-by-n counter.

The modulus of a counter is given as: 2^n where n = number of flip-flops. So a 3 flip-flop counter will have a maximum count of $2^3 = 8$ counting states and would be called a MOD-8 counter. The maximum binary number that can be counted by the counter is $2^n–1$ giving a maximum count of $(111)_2 = 2^3–1 = 7_{10}$. Then the counter counts from 0 to 7.

Common MOD counters include those with MOD numbers of 2, 4, 8 and 16 and with the use of external combinational circuits can be configured to count to any predetermined value other than

one with a maximum 2^n modulus. In general, any arrangement of a "m" number of flip-flops can be used to construct any MOD counter.

A common modulus for counters with truncated sequences is ten (1010), called MOD-10. A counter with ten states in its sequence is known as a decade counter. Decade counters are useful for interfacing to digital displays. Other MOD counters include the MOD-6 or MOD-12 counter which have applications in digital clocks to display the time of day.

A Synchronous Counter Design Using D Flip-Flops and J-K Flip-Flops

(A Synchronous Counter Design, 2016)

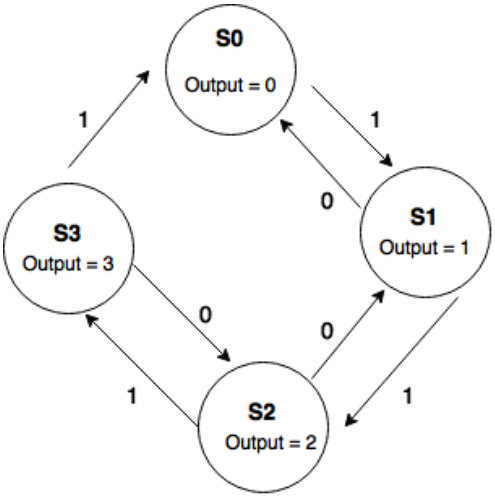
For this project, I will show how to design a synchronous counter which is capable of storing data and counting either up or down, based on input, using either D flip-flops or J-K flip-flops. Specifically, the counter will count up: 0, 1, 2, 3, 0, 1, 2, 3, … when the input x = 1, and count down when the input x = 0.

Suggested state definition tables, transition diagrams, transition tables, K-maps for the respective logic functions, and schematics of the implementation using flipflops and logic gates for both a D flip-flop and a J-K flip-flop scenario will be given.

DESIGN #1 – Synchronous Counter: D Flip-Flops

State	Definition	Binary values
S0	Reset/Initialize, no sequence	00
S1	Count 1	01
S2	Count 2	10
S3	Count 3	11

Figure 1: State Transition Diagram (D Flip-Flops)



Synchronous Counter Counter D Flip Flop

Table 2: State Transition Table (D Flip-Flops)

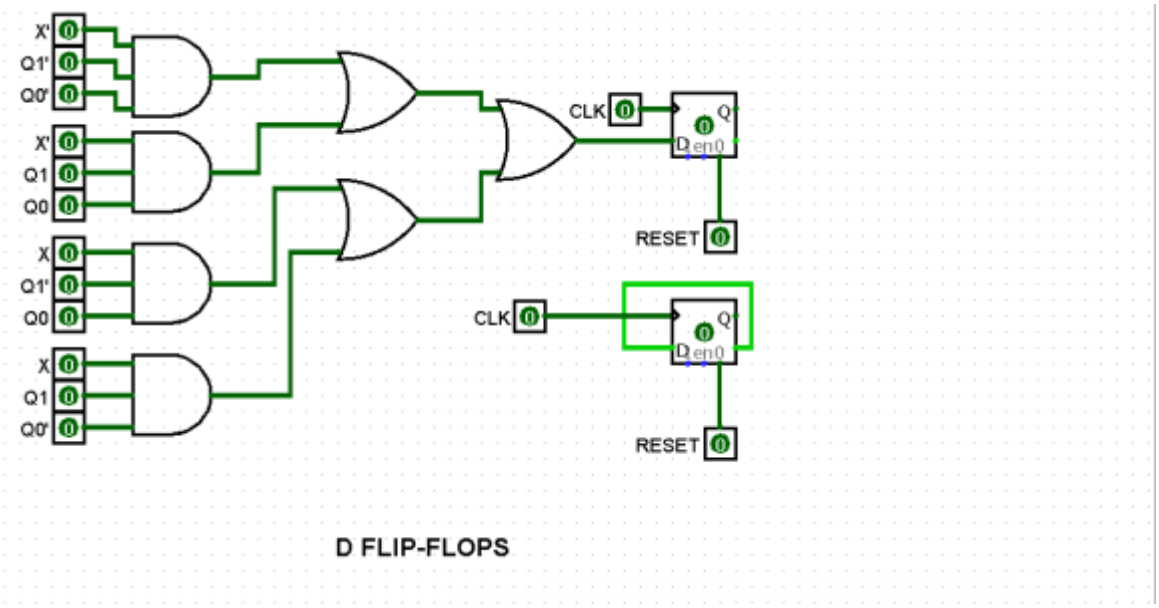
Input	Present State			Next State			FF Inputs	
x	State	Q_1	Q_0	State	Q_1^+	Q_0^+	D_1	D_0
0	S_0	0	0	S_3	1	1	1	1
0	S_1	0	1	S_0	0	0	0	0
0	S_2	1	0	S_1	0	1	0	1
0	S_3	1	1	S_2	1	0	1	0
1	S_0	0	0	S_1	0	1	0	1
1	S_1	0	1	S_2	1	0	1	0
1	S_2	1	0	S_3	1	1	1	1
1	S_3	1	1	S_0	0	0	0	0
Synchronous Counter – State Transition Using D Flip-Flops								

Tables 3 and 4: K-Maps for D Flip-Flop Design

x \ Q_1Q_0	**00**	**01**	**11**	**10**
0	1	0	1	0
1	0	1	0	1
Synchronous Counter – (D_1) K-Map				
$D_1 = x'Q_1'Q_0' + x'Q_1Q_0 + xQ_1'Q_0 + xQ_1Q_0'$				

x \ Q_1Q_0	**00**	**01**	**11**	**10**
0	1	0	0	1
1	1	0	0	1
Synchronous Counter – (D_0) K-Map				
$D_0 = Q_0'$				

Schematic of D Flip-Flop Using Logisim Software [6]:

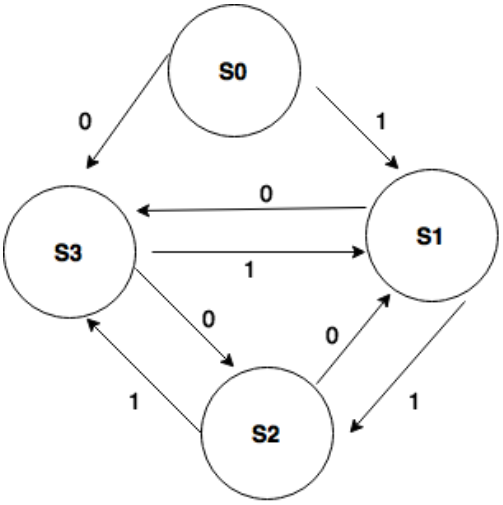


DESIGN #2 – SYNCHRONOUS COUNTER: J-K FLIP-FLOPS

Table 5: State Definition Table (Design 2 – J-K Flip Flops)

State	Definition	Binary values
S0	Reset/Initialize, no sequence	00
S1	Count 1	01
S2	Count 2	10
S3	Count 3	11

Figure 4: State Transition Diagram (J-K Flip-Flops)



Synchronous Counter Counter JK Flip Flop

Table 6: State Transition Table (J-K Flip-Flops)

Input	Present State			Next State			JK Inputs			
x	**State**	**Q_1**	**Q_0**	**State**	**Q_1^+**	**Q_0^+**	**J_1**	**K_1**	**J_0**	**K_0**
0	S_0	0	0	S_3	1	1	1	X	1	X
0	S_1	0	1	S_0	1	1	1	X	X	0
0	S_2	1	0	S_1	0	1	X	1	1	X
0	S_3	1	1	S_2	1	0	X	0	X	1
1	S_0	0	0	S_1	0	1	0	X	1	X
1	S_1	0	1	S_2	1	0	1	X	X	1
1	S_2	1	0	S_3	1	1	X	0	1	X
1	S_3	1	1	S_0	0	1	X	1	X	0
Synchronous Counter – State Transition Using J-K Flip-Flops										

Tables 7-10: K-Maps for J-K Flip-Flop Design

x \ Q_1Q_0	**00**	**01**	**11**	**10**
0	1	1	X	X
1	0	1	X	X
Synchronous Counter J-K Flip Flops – (J_1) K-Map				
$J_1 = x' + Q_0$				

I hope you find these project results interesting and useful, as well as a good demonstration of how powerful these relatively simple circuits can be. If you would like more information, I recommend the text by Alan B. Marcovitz, *Introduction to Logic Design*. [7] You can also download the Logisim digital-logic simulation software for free, at: http://www.cburch.com/logisim/ [8].

Reference

A Synchronous Counter Design. (2016). Retrieved from K.L. Cratt - Website and Blog: https://klcraft.net/2017/06/21/a-synchronous-counter-design-using-a-d-flip-flop-and-a-j-k-flip-flop/

[1] V. Pedroni, *Digital Electronics and Design with VHDL*, Morgan Kaufmann (2008).

[2] A.M. Niknejad, *Latches and Flip Flops*, University of California-Berkeley (2010). Available: http://rfic.eecs.berkeley.edu/ee100/pdf/lect24.pdf

[3] J.P. Eckert, *A Survey of Digital Computer Memory Systems*, Proceedings of the IRE, Vol. 41: Issue 10, IEEE Explore, Available: http://ieeexplore.ieee.org/document/4051207/

[4] K. Bigelow, *The S Flip-Flop* (2017), Available: http://www.play-hookey.com/digital/sequential/d_nand_flip-flop.html