Lab 3 – Sequence circuits

135 mins / 3 lectures.

Main objects

A. Clock rate
B. SR Flip flops
C. D Flip flops.
D. JK Flip flops.
E. Counters.

Resources

Object	Software / File	Book	Slide	Student's task

Preferences

[1] Brian Holdsworth, Clive Woods, [2002], Digital Logic Design, 4th edition, Newnes, Oxford.

[2] https://www.electronics-tutorials.ws/combination/ access on Jan 14th.

Table of Contents

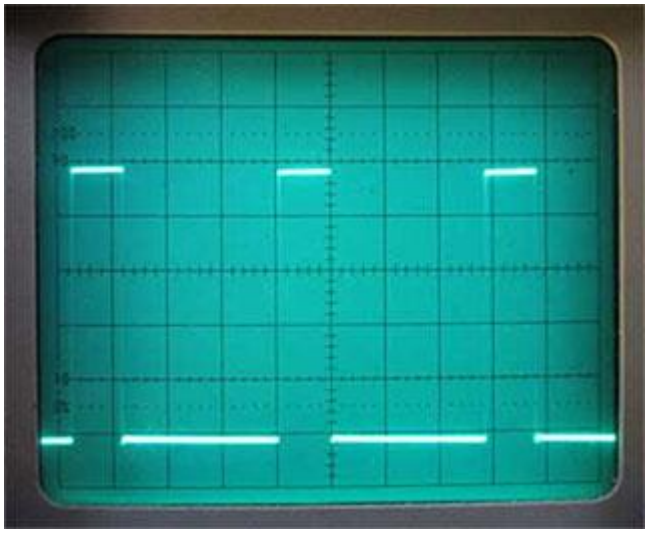
Part 1. Sequence circuit

The logic circuits discussed in LAB 2 had output states that depended on the particular combination of logic states at the input connections to the circuit. For this reason these circuits are called combinational logic circuits.

This LAB tutorial introduces digital circuits that use SEQUENTIAL LOGIC.

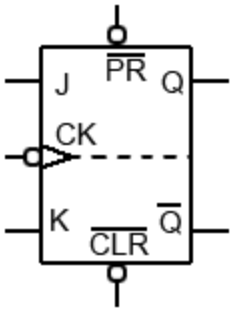
In these circuits the output depends, not only on the combination of logic states at its inputs, but also on the logic states that existed previously. In other words the output depends on a SEQUENCE of events occurring at the circuit inputs. Examples of such circuits include clocks, flip-flops, bi-stables, counters, memories, and registers. The actions of these circuits depend on a range of basic sub-circuits.

Clock Circuits

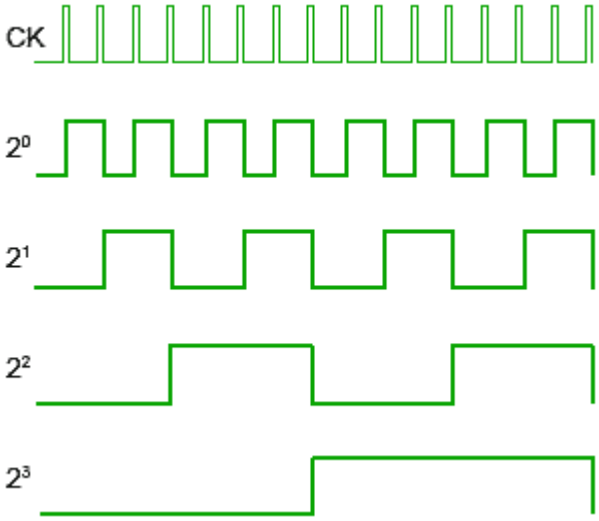


Module 5.1 deals with clock oscillators, which are basically types of square wave generators or oscillators that produce a continuous stream of square waves or a continuous train of pulses (a "square" wave whose mark to space ratio is NOT 1:1). These pulses are used to sequence the actions of other devices in the sequential logic circuit so that all the actions taking place in the circuit are properly synchronised.

Bi-Stable Logic Devices

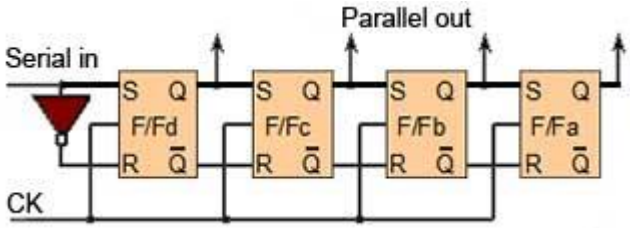


Bi-stable devices (popularly called Flip-flops) described in Modules 5.2 to 5.4, are sub-circuits, usually contained within ICs, and are the most basic type of 1-bit memory. They have outputs that can take up one of two stable states, Logic 1 or logic 0 or off. Once the device is triggered into one of these two states by an external input pulse, the output remains in that state until another pulse is used to reverse that state, so that a logic 1 output becomes logic 0 or vice versa. Again the circuit remains stable in this state until an input signal is used to reverse the output state. Hence the circuit is said to have Bi (two) stable output states.



Counters

Various types of digital counters are described in next chapters. Consisting of arrangements of bi-stables, they are very widely used in many types of digital systems from computer arithmetic to TV screens, as well as many digital timing and measurement devices.



Registers

Also consisting of arrays of bi-stable elements, the shift registers described in Module 5.7 are temporary storage devices (memories) for multi-bit digital data. The data can be stored in the register either one bit at a time (serial input) or as one or more bytes at a time (parallel input).

The register can then output the data in either serial or parallel form. Shift registers are vital to receiving or transmitting data in digital

communications systems. They can also be used in digital arithmetic for operations such as multiplication and division.

A Simple ALU

A simple arithmetic and logic unit (ALU) is described in Module 5.8 and combines many of the combinational and sequential logic circuits described in modules 4 and 5 to demonstrate how a very complex application is built by combining a number of much simpler digital sub circuits.

Part 2. Clocks and Timing Signals

Most sequential logic circuits are driven by a clock oscillator. This usually consists of an astable circuit producing regular pulses that should ideally:

1. Be constant in frequency

Many clock oscillators use a crystal to control the frequency. Because crystal oscillators generate normally high frequencies, where lower frequencies are required the original oscillator frequency is divided down from a very high frequency to a lower one using counter circuits.

2. Have fast rising and falling edges to its pulses.

It is the edges of the pulses that are important in timing the operation of many sequential circuits, the rise and fall times are usually be less than 100ns. The outputs of clock circuits will typically have to drive more gates than any other output in a given system. To prevent this load distorting the clock signal, it is usual for clock oscillator outputs to be fed via a buffer amplifier.

3. Have the correct logic levels

The signals produced by the clock circuits must have appropriate the logic levels for the circuits being supplied.

Simple Clock Oscillator

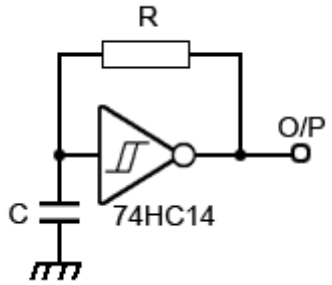


Figure 1. Simple Schmitt Inverter Clock Oscillator

Crystal Controlled Clock Oscillator

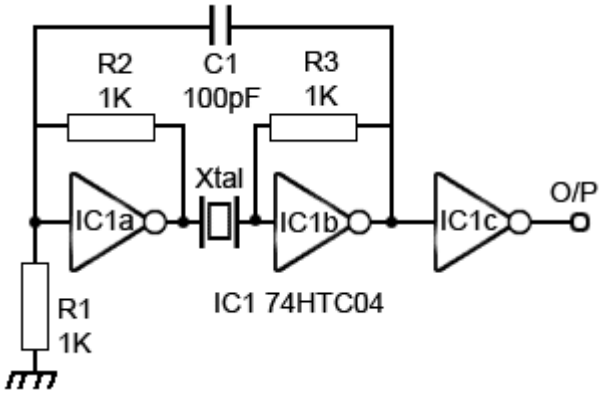


Figure 2. Crystal Controlled Clock Oscillator

Clock in Logisim

The clock toggles its output value on a regular schedule as long as ticks are enabled via the Simulate menu. (Ticks are disabled by default.) A "tick" is Logisim's unit of time; the speed at which ticks occur can be selected from the Simulate menu's Tick Frequency submenu.

The clock's cycle can be configured using its High Duration and Low Duration attributes.

Note that Logisim's simulation of clocks is quite unrealistic: In real circuits, multiple clocks will drift from one another and will never move in lockstep. But in Logisim, all clocks experience ticks at the same rate.

A clock has only one pin, an output with a bit width of 1, whose value will represent the current value of the clock. The location of this pin is specified in the Facing attribute. The clock's value will toggle on its schedule whenever ticks are enabled, and it will toggle whenever it is clicked using the Poke Tool.

Ex 1: Using a clock generator, output to a LED and setting to vary frequencies, experiment the LED.

- *Set frequency to 1Hz, 4Hz, and more …*
- *Set the clock to 4 tick LOW and 1 tick HIGH.*

Part 3. SR Flip-flops

Typical applications for SR Flip-flops.

The basic building bock that makes computer memories possible, and is also used in many sequential logic circuits is the flip-flop or bi-stable circuit. Just two inter-connected logic gates make up the basic form of this circuit whose output has two stable output states. When the circuit is triggered into either one of these states by a suitable input pulse, it will 'remember' that state until it is changed by a further input pulse, or until power is removed. For this reason, the circuit may also be called a Bi-stable Latch.

The SR flip-flop can be considered as a 1-bit memory, since it stores the input pulse even after it has passed. Flip-flops (or bi-stables) of different types can be made from logic gates and, as with other combinations of logic gates, the NAND and NOR gates are the most versatile, the NAND being most widely used. This is because, as well as being universal, i.e. it can be made to mimic any of the other standard logic functions, it is also cheaper to construct. Other, more widely used types of flip-flop are the JK, the D type and T type, which are developments of the SR flip-flop and will be studied later.

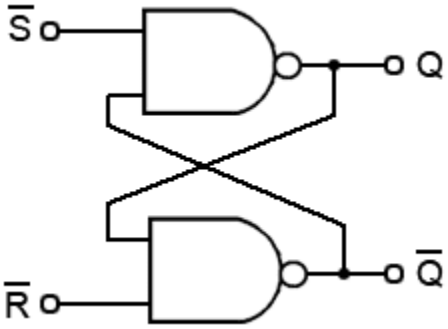


Figure 3. SR Flip-flop (low activated)

The SR Flip-flop.

The SR (Set-Reset) flip-flop is one of the simplest sequential circuits and consists of two gates connected as shown in Fig. 3. Notice that the output of each gate is connected to one of the inputs of the other gate, giving a form of positive feedback or 'cross-coupling'.

The circuit has two active low inputs marked S and R, 'NOT' being indicated by the bar above the letter, as well as two outputs, Q and Q. Table 5.2.1 shows what happens to the Q and Q outputs when a logic 0 is applied to either the S or R inputs.

Table 5.2.1					
	$\overline{S}$	$\overline{R}$	Q	$\overline{Q}$	Comments
1.	0	1	1	0	Q is set to 1 by 0 on $\overline{S}$
2.	1	1	1	0	No change, (1 on Q is remembered)
3.	1	0	0	1	Q is reset to 0 by 0 on $\overline{R}$
4.	1	1	0	1	No change, (0 on Q is remembered)
5.	0	0	1	1	Both inputs at 0 – both outputs are at 1 (Non-allowed state)
6.	1	1	?	?	Inputs change from 0,0 to 1,1 together - outputs will be INDETERMINATE

The SR Flip-flop Truth Table (Table 5.2.1)

1. Q output is set to logic 1 by applying logic 0 to the S input.
2. Returning the S input to logic 1 has no effect. The 0 pulse (high-low-high) has been 'remembered' by the Q.
3. Q is reset to 0 by logic 0 applied to the R input.
4. As R returns to logic 1 the 0 on Q is 'remembered' by Q.

Ex 2: Build a Set-Reset Flip Flop as a sub circuit and test it.

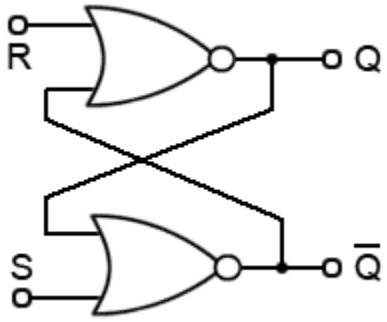
?

- *Build by diagram above.*
- *Could we build from truth table?*
- *Notice the "negative" inputs and outputs.*

The RS Latch

Flip-flops can also be considered as latch circuits due to them remembering or 'latching' a change at their inputs. A common form of RS latch is shown in Fig. 5.2.5. In this circuit the S and R inputs have now become S and R inputs, meaning that they will now be 'active high'.

They have also changed places, the R input is now on the gate having the Q output and the S input is on the Q gate. These changes occur because the circuit is using NOR gates instead of NAND.



RS Latch Truth Table (Table 5.2.2)

Table 5.2.2					
	R	S	Q	$\overline{Q}$	Comments
1.	0	1	1	0	Q is set to 1 by 1 on S
2.	0	0	1	0	No Change after set
3.	1	0	0	1	Q is reset to 0 by 1 on R
4.	0	0	0	1	No Change after reset
5.	1	1	0	0	Not allowed (both outputs at 0)
6.	0	0	?	?	If both inputs change from 11 to 00 outputs will be INDETERMINATE

1. Q is set to 1 when the S input goes to logic 1.
2. This is remembered on Q after the S input returns to logic 0.
3. Q is reset set to 0 when the R input goes to logic 1.
4. This is remembered on Q after the R input returns to logic 0.
5. If both inputs are at logic 1, Q is the same as Q (the non-allowed state).
6. The state of the outputs cannot be guaranteed if the inputs change from 1,1 to 0, 0 at the same time.

Ex 3: Build a Set-Reset Latch as a sub circuit and test it.

- *Build by diagram above.*
- *Could we build from truth table?*
- *What is the difference between Flip-flop and Latch?*

Timing Diagrams

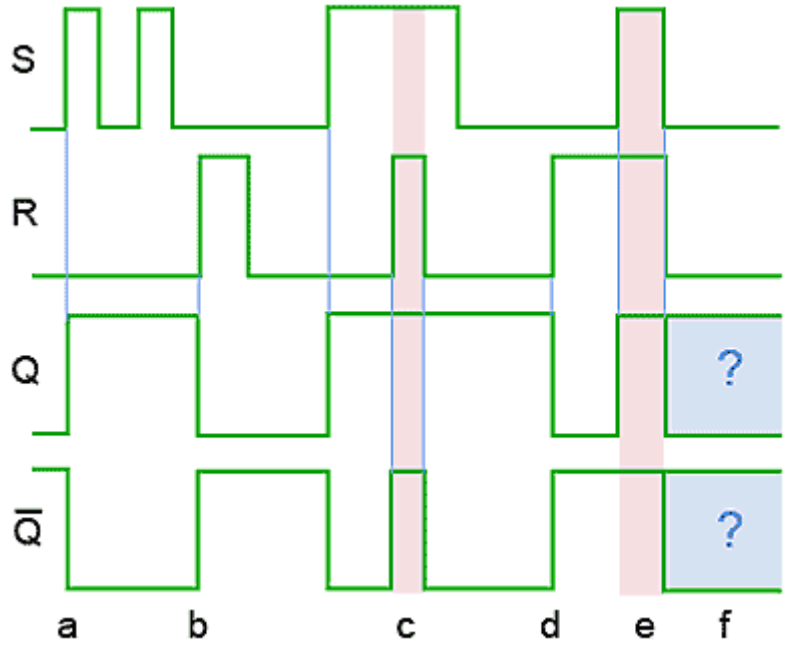


Figure 4. RS Latch Timing Diagram

Truth tables are not always the best method for describing the action of a sequential circuit such as the SR flip-flop. Timing diagrams, which show how the logic states at various points in a circuit vary with time, are often preferred.

The Clocked SR Flip-flop

By adding two extra NAND gates, the timing of the output changeover after a change of logic states at S and R can be controlled by applying a logic 1 pulse to the clock (CK) input. Note that the inputs are now labelled S and R indicating that the inputs are now ‘high activated’. This is because the two extra NAND gates are disabled while the CK input is low, therefore the outputs are completely isolated from the inputs and so retain any previous logic state, but when the CK input is high (during a clock pulse) the input NAND gates act as inverters. Then for example, a logic 1 applied to S becomes a logic 0 applied to the Sinput of the active low SR flip-flop second stage circuit.

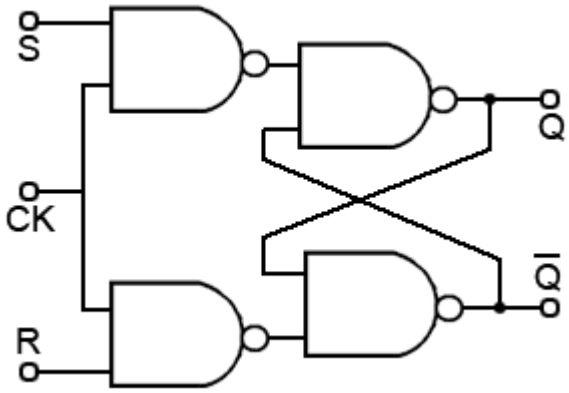


Figure 5. High Activated Clocked SR Flip-flop

The main advantage of the CK input is that the output of this flip-flop can now be synchronised with many other circuits or devices that share the same clock. This arrangement could be used for a basic memory location by, for example, applying different logic states to a range of 8 flip-flops, and then applying a clock pulse to CK to cause the circuit to store a byte of data.

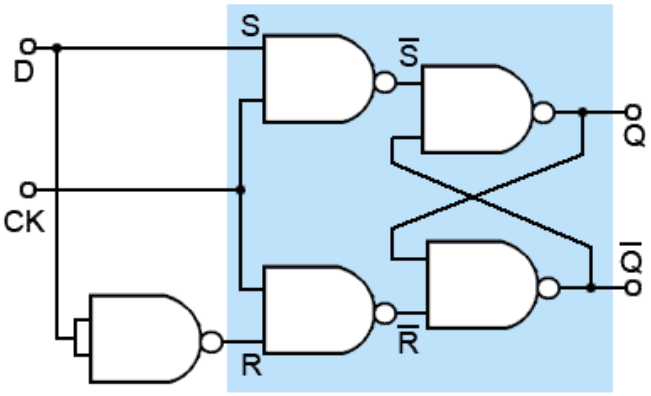
The basic form of the clocked SR flip-flop shown in Fig. 5.2.7 is an example of a level triggered flip-flop. This means that outputs can only change to a new state during the time that the clock pulse is at its high level (logic 1). The ability to change the input whilst CK is high can be a problem with this circuit, as any input changes occurring during the high CK period, will also change the outputs. A better method of triggering, which will only allow the outputs to change at one precise instant is provided by edge triggered devices available in D Type and JK flip-flops.

Ex 4: Build a Clocked Set-Reset Latch as a sub circuit and test it.

?

- *Build by diagram above.*
- *Could we build from truth table?*
- *What is the advantage of "Clocked control"?*

Part 4. D Flip Flop



Inputs		Outputs	
CK	D	Q	$\overline{Q}$
0	X	No change	
1	0	0	1
1	1	1	0

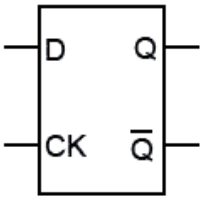


Fig. 5.3.1 Level Triggered D Type Flip-flop

D Type Flip-flops

The major drawback of the SR flip-flop (i.e. its indeterminate output and non-allowed logic states) described in previous chapter is overcome by the D type flip-flop. This flip-flop, shown above together with its truth table and a typical schematic circuit symbol, may be called a Data flip-flop because of its ability to 'latch' and remember data, or a Delay flip-flop because latching and remembering data can be used to create a delay in the progress of that data through a circuit. To avoid the ambiguity in the title therefore, it is usually known simply as the D Type. The simplest form of D Type flip-flop is basically a high activated SR type with an additional inverter to ensure that the S and R inputs cannot both be high or both low at the same time. This simple modification prevents both the indeterminate and non-allowed states of the SR flip-flop. The S and R inputs are now replaced by a single D input, and all D type flip-flops have a clock input.

Operation.

As long as the clock input is low, changes at the D input make no difference to the outputs. The truth table in Fig. 5.3.1 shows this as a 'don't care' state (X). The basic D Type flip-flop shown in Fig. 5.3.1 is called a level triggered D Type flip-flop because whether the D input is active or not depends on the logic level of the clock input.

Provided that the CK input is high (at logic 1), then whichever logic state is at D will appear at output Q and (unlike the SR flip-flops) Q is always the inverse of Q).

In Fig. 5.3.1, if D = 1, then S must be 1 and R must be 0, therefore Q is SET to 1.

Alternatively,

If D = 0 then R must be 1 and S must be 0, causing Q to be reset to 0.

? **Ex 5: Build a D Latch as a sub circuit and test it.**

Synchronous and Asynchronous Inputs

A further refinement is the addition of two further inputs SET and RESET, which are actually the original Sand R inputs of the basic low activated SR flip-flop.

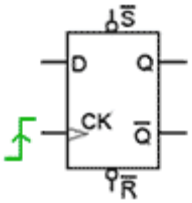


Figure 6. Edge Triggered D Type Flip-Flop

Notice that there is now a subtle difference between the active low Set (S) and Reset (R) inputs, and the D input. The D input is SYNCHRONOUS, that is its action is synchronised with the clock, but the S and R inputs are ASYNCHRONOUS i.e. their action is NOT synchronised with the clock. The SET and RESET inputs in Fig 5.3.4 are 'low activated', which is shown by the inversion circles at the S and R inputs to indicate that they are really S and R.

The flip-flop is positive edge triggered, which is shown on the CK input in Fig 5.3.4 by the wedge symbol. A wedge accompanied by an inversion circle would indicate negative (falling) edge triggering, though this is generally not used on D Type flip-flops.

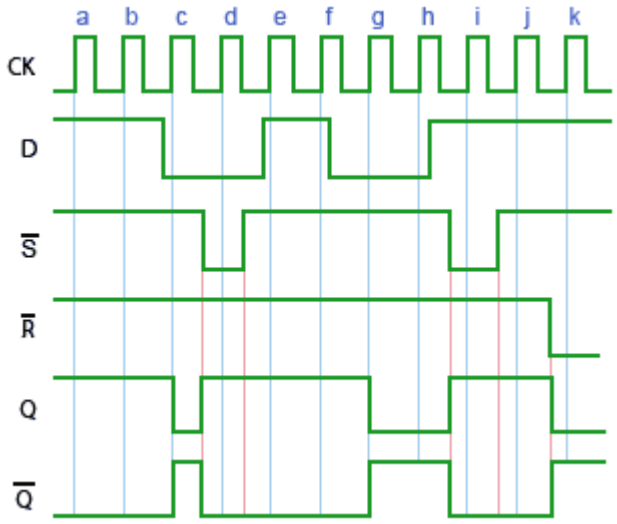


Figure 7. Typical Schematic Symbols for D Type Edge Triggered Flip-Flops

Ex 6: Build a D Latch with Set/Reset inputs as a sub circuit and test it.

The Toggle Flip-flop

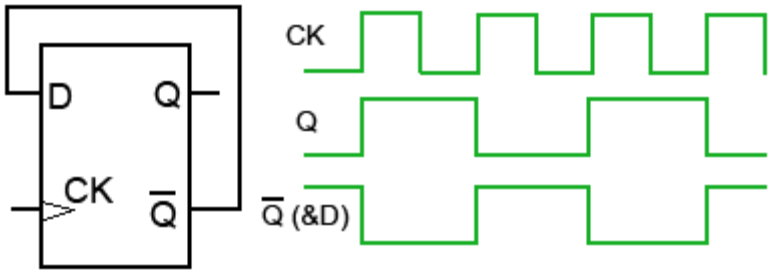


Figure 8. An Edge Triggered D Type Converted to a Toggle Flip-flop

Toggle flip-flops are the basic components of digital counters, and all of the D type devices are adaptable for such use. When an electronic counter is used for counting, what are actually being counted are pulses appearing at the CK input, which may be either regular pulses derived from an internal clock, or they can be irregular pulses generated by some external event.

When a toggle flip-flop is used as one stage of a counter, its Q output changes to the opposite state, (it toggles) high or low on each clock pulse. Most edge-triggered flip-flops can be used as toggle flip-flops including the D type, which can be converted to a toggle flip-flop with a simple modification. In theory all that is necessary to convert an edge triggered D Type to a T type is to connect the Q output directly to the D input as shown in Fig. 5.3.8. The actual input is now CK. The effect of this mode of operation is also shown in the timing diagram in Fig. 5.3.8 using a positive edge triggered D type flip-flop.

Ex 7: Build a Toggle filp flop as a sub circuit and test it.

- *Build by diagram above.*
- *After "toggle" the frequency is divided by 2.*
- *Connect some Toogle flip flops together and test it.*

Part 5. JK Flip-flops

A Universal Programmable Flip-flop

The JK Flip-flop is also called a programmable flip-flop because, using its inputs, J, K, S and R, it can be made to mimic the action of any of the other flip-flop types.

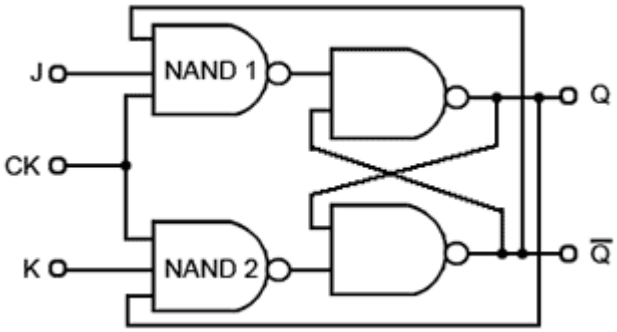


Figure 9. Basic JK Flip-flop Circuit

Ex 7: Build a J-K filp flop as a sub circuit and test it.

- *Build by diagram above.*

Operation

As a starting point, assume that both J and K are at logic 1 and the outputs Q = 0 and Q = 1, this will cause NAND 1 to be enabled, as it has logic 1 on two (J and Q) of its three inputs, requiring only a logic 1 on its clock input to change its output state to logic 0. At the same time, NAND 2 is disabled, because it only has one of its inputs (K) at logic 1, its feedback input is at logic 0 because of the feedback from Q.

On the arrival of a clock pulse, the output of NAND 1 therefore becomes logic 0, and causes the flip-flop to change state so that Q = 1 and Q = 0. This action enables NAND 2 and disables NAND 1.

JK Synchronous Inputs

Table 5.4.1				
J	K	Q Before CK Pulse	Q After CK Pulse	Comments
0	0	0	0	No Change
0	0	1	1	
1	0	0	1	$Q = J$ $\overline{Q} = K$
1	0	1	1	
0	1	0	0	
0	1	1	0	
1	1	0	1	Outputs Toggle
1	1	1	0	

• When J and K are both 0 the flip-flop is inhibited, Q is the same after the CK pulse as it was before; there is no change at the output.

• If J and K are at different logic levels, then after the CK pulse, Q and Q will take up the same states as J and K. For example, if J = 1 and K = 0, then on the trailing (negative going) edge of a clock pulse, the Q output will be set to 1, and if K = 1 and J = 0 then the Q output is reset to logic 0 on the trailing edge of a clock pulse, effectively mimicking **the D type master slave flip-flop** by replacing the D input with J.

• If logic 1 is applied to both J and K, the output toggles at the trailing edge of each clock pulse, just like a toggle flip-flop.

The JK flip-flop can therefore be called a 'programmable flip-flop' because of the way its action can be programmed by the states of J and K.

Each of the above actions are synchronised with the clock pulse, data being taken into the master flip-flop at the rising edge of the clock pulse, and output from the slave flip-flop appears at the falling edge of the clock pulse.

Note: Although the above describes the action of a master slave JK flip-flop, there are also positive edge and negative edge triggered versions available.

Asynchronous Inputs

Asynchronous inputs, which act independently of the clock pulse, are also provided by the active low inputs PR and CLR. These act as (usually active low) SET and RESET inputs respectively, and as they act independently of the clock input, they give the same facilities as a simple SR flip-flop. As with the SR flip-flop, in this mode some external method is needed to ensure that these two inputs cannot both be active at the same time, as this would make both Q and Q logic 1.

Ex 8: Build a J-K filp flop with asynchronous inputs as a sub circuit and test it.

- *Build by diagram above.*

Part 6. Digital Counters

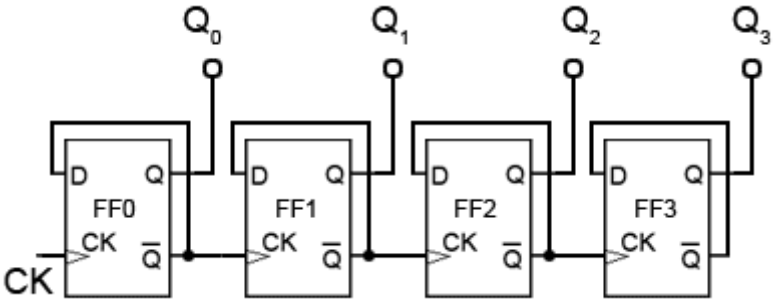


Figure 10. Four-bit Asynchronous Up Counter

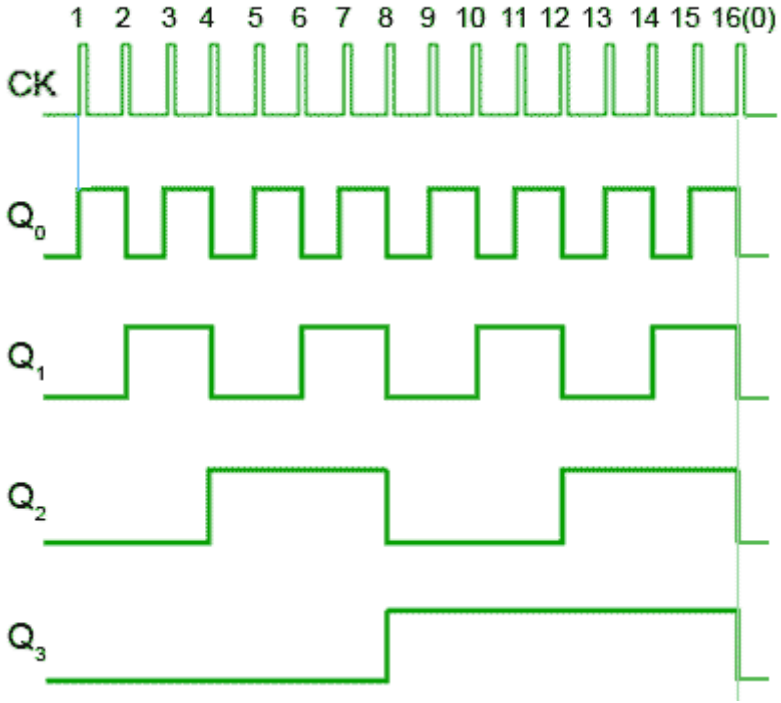


Figure 11. Four-bit Asynchronous Up Counter Waveforms

Asynchronous Counters.

Counters, consisting of a number of flip-flops, count a stream of pulses applied to the counter's CK input. The output is a binary value whose value is equal to the number of pulses received at the CK input.

Each output represents one bit of the output word, which, in 74 series counter ICs is usually 4 bits long, and the size of the output word depends on the number of flip-flops that make up the counter. The output lines of a 4-bit counter represent the values 2^0, 2^1, 2^2 and 2^3, or 1,2,4 and 8 respectively. They are normally shown in schematic diagrams in reverse order, with the least significant bit at the left, this is to

enable the schematic diagram to show the circuit following the convention that signals flow from left to right, therefore in this case the CK input is at the left.

Four Bit Asynchronous Up Counter

Fig. 5.6.1 shows a 4 bit asynchronous up counter built from four positive edge triggered D type flip-flops connected in toggle mode. Clock pulses are fed into the CK input of FF0 whose output, Q_0 provides the 2^0 output for FF1 after one CK pulse.

The rising edge of the Q output of each flip-flop triggers the CK input of the next flip-flop at half the frequency of the CK pulses applied to its input.

The Q outputs then represent a four-bit binary count with Q_0 to Q_3 representing 2^0 (1) to 2^3 (8) respectively.

Assuming that the four Q outputs are initially at 0000, the rising edge of the first CK pulse applied will cause the output Q_0 to go to logic 1, and the next CK pulse will make Q_0 output return to logic 0, and at the same time Q_0 will go from 0 to 1.

As Q_0 (and the CK input of FF1 goes high) this will now make Q_1 high, indicating a value of 2^1 (2_{10}) on the Q outputs.

The next (third) CK pulse will cause Q_0 to go to logic 1 again, so both Q_0 and Q_1 will now be high, making the 4-bit output 1100_2 (3_{10} remembering that Q_0 is the least significant bit).

The fourth CK pulse will make both Q_0 and Q_1 return to 0 and as Q_1 will go high at this time, this will toggle FF2, making Q_2 high and indicating 0010_2 (4_{10}) at the outputs.

Reading the output word from right to left, the Q outputs therefore continue to represent a binary number equalling the number of input pulses received at the CK input of FF0. As this is a four-stage counter the flip-flops will continue to toggle in sequence and the four Q outputs will output a sequence of binary values from 0000_2 to 1111_2 (0 to 15_{10}) before the output returns to 0000_2 and begins to count up again as illustrated by the waveforms in Fig 5.6.2.

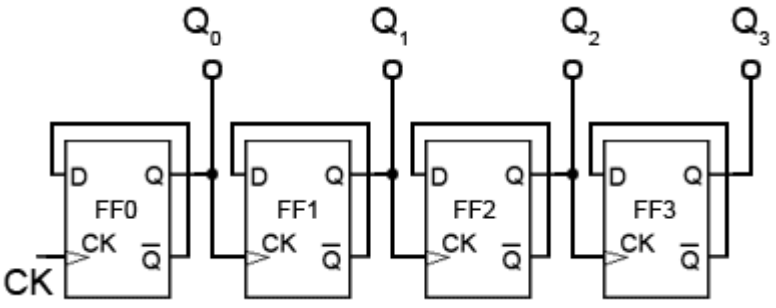


Fig. 5.6.3 Four-bit Asynchronous Down Counter

Four Bit Asynchronous Down Counter

To convert the up counter in Fig. 5.6.1 to count DOWN instead, is simply a matter of modifying the connections between the flip-flops. By taking both the output lines and the CK pulse for the next flip-flop in sequence from the Q output as shown in Fig. 5.6.3, a positive edge triggered counter will count down from 1111_2 to 0000_2.

Although both up and down counters can be built, using the asynchronous method for propagating the clock, they are not widely used as counters as they become unreliable at high clock speeds, or when a large number of flip-flops are connected together to give larger counts, due to the clock ripple effect.

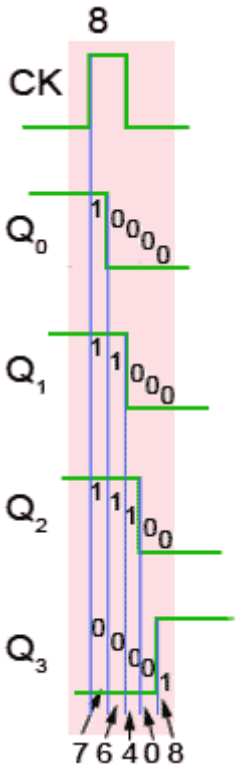


Fig.5.6.4 Timing Diagram Detail Showing Clock Ripple

Clock Ripple

The effect of clock ripple in asynchronous counters is illustrated in Fig. 5.6.4, which is a magnified section (pulse 8) of Fig. 5.6.2.

Fig. 5.6.4 shows how the propagation delays created by the gates in each flip-flop (indicated by the blue vertical lines) add, over a number of flip-flops, to form a significant amount of delay between the time at which the output changes at the first flip flop (the least significant bit), and the last flip flop (the most significant bit).

As the Q_0 to Q_3 outputs each change at different times, a number of different output states occur as any particular clock pulse causes a new value to appear at the outputs.

At CK pulse 8 for example, the outputs Q_0 to Q_3 should change from 1110_2 (7_{10}) to 0001_2 (8_{10}), however what really happens (reading the vertical columns of 1s and 0s in Fig. 5.6.4) is that the outputs change, over a period of around 400 to 700ns, in the following sequence:

$1110_2 = 7_{10}$

$0110_2 = 6_{10}$

$0010_2 = 4_{10}$

$0000_2 = 0_{10}$

$0001_2 = 8_{10}$

At CK pulses other that pulse 8 of course, different sequences will occur, therefore there will be periods, as a change of value ripples through the chain of flip-flops, when unexpected values appear at the Q outputs for a very short time. However this can cause problems when a particular binary value is to be selected, as in the case of a decade counter, which must count from 0000_2 to 1001_2 (9_{10}) and then reset to 0000_2 on a count of 1010_2 (10_{10}).

These short-lived logic values will also cause a series of very short spikes on the Q outputs, as the propagation delay of a single flip-flop is only about 100 to 150ns. These spikes are called ‘runt spikes’ and although they may not all reach to full logic 1 value every time, as well as possibly causing false counter triggering, they must also be considered as a possible cause of interference to other parts of the circuit.

Although this problem prevents the circuit being used as a reliable counter, it is still valuable as a simple and effective frequency divider, where a high frequency oscillator provides the input and each flip-flop in the chain divides the frequency by two.

Synchronous Counters

The synchronous counter provides a more reliable circuit for counting purposes, and for high-speed operation, as the clock pulses in this circuit are fed to every flip-flop in the chain at exactly the same time. Synchronous counters use JK flip-flops, as the programmable J and K inputs allow the toggling of individual flip-flops to be enabled or disabled at various stages of the count. Synchronous counters therefore eliminate the clock ripple problem, as the operation of the circuit is synchronised to the CK pulses, rather than flip-flop outputs.

Synchronous Up Counter

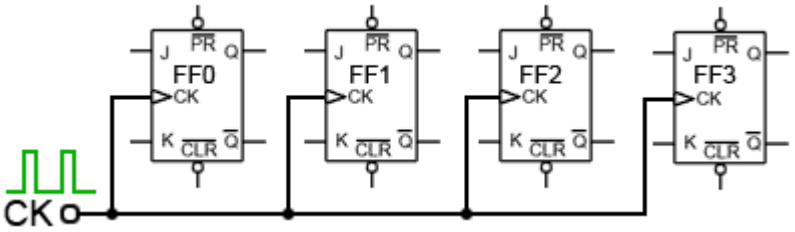


Figure 12. Synchronous Clock Connection

Fig. 5.6.5 shows how the clock pulses are applied in a synchronous counter. Notice that the CK input is applied to all the flip-flops in parallel. Therefore, as all the flip-flops receive a clock pulse at the same instant, some method must be used to prevent all the flip-flops changing state at the same time. This of course would result in the counter outputs simply toggling from all ones to all zeros, and back again with each clock pulse.

However, with JK flip-flops, when both J and K inputs are logic 1 the output toggles on each CK pulse, but when J and K are both at logic 0 no change takes place.

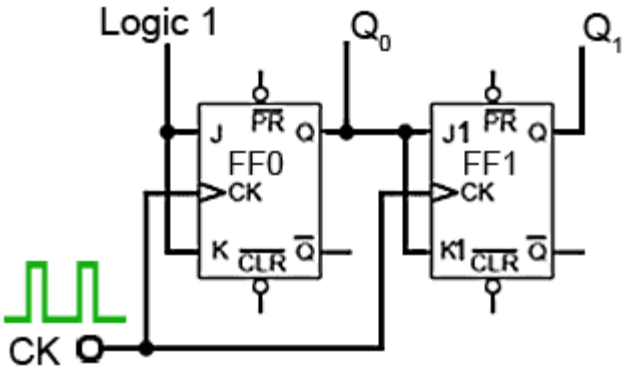


Fig. 5.6.6 The First Two Stages of a Synchronous Counter

Fig. 5.6.6 shows two stages of a synchronous counter. The binary output is taken from the Q outputs of the flip-flops. Note that on FF0 the J and K inputs are permanently wired to logic 1, so Q_0 will change state (toggle) on each clock pulse. This provides the ‘ones’ count for the least significant bit.

On FF1 the J1 and K1 inputs are both connected to Q_0so that FF1 output will only be in toggle mode when $Q_0$is also at logic 1. As this only happens on alternate clock pulses, Q_1 will only toggle on even numbered clock pulses giving a ‘twos’ count on the Q_1 output.

Table 5.6.1					
CK	Q_0	Q_1	J1	K1	After the CK pulse
0	0	0	0	0	No CK pulses yet
1	1	0	1	1	Q_0 toggles making J1 & K1 = 1
2	0	1	0	0	Q_0 & Q_1 toggle making J1 & K1 = 0
3	1	1	1	1	Only Q_0 toggles making J1 & K1 = 1
4	0	0	0	0	Q_0 & Q_1 toggle making J1 & K1 = 0

Table 5.6.1 shows this action, where it can be seen that Q_1 toggles on the clock pulse only when J1 and K1 are high, giving a two bit binary count on the Q outputs, (where Q_0 is the least significant bit).

In adding a third flip flop to the counter however, direct connection from J and K to the previous Q_1 output would not give the correct count. Because Q_1 is high at a count of 2_{10} this would mean that FF2 would toggle on clock pulse three, as J2 and K2 would be high. Therefore clock pulse 3 would give a binary count of 111_2 or 7_{10} instead of 4_{10}.

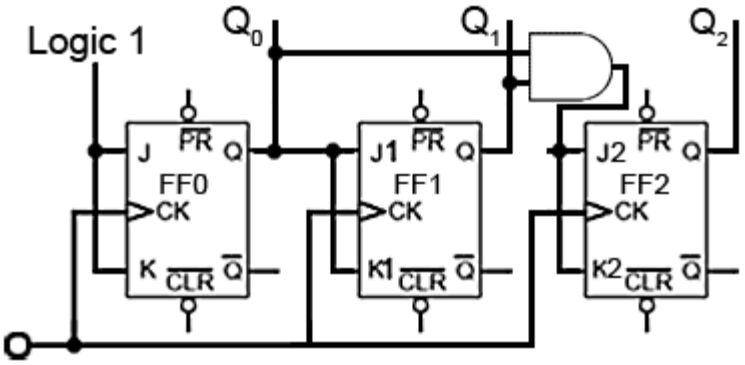


Fig. 5.6.7 Adding a Third Stage

To prevent this problem an AND gate is used, as shown in Fig. 5.6.7 to ensure that J2 and K2 are high only when both Q_0 and Q_1 are at logic 1 (i.e. at a count of three). Only when the outputs are in this state will the next clock pulse toggle Q_2 to logic 1. The outputs Q_0 and Q_1 will of course return to logic 0 on this pulse, so giving a count of 001_2 or 4_{10} (with Q_0 being the least significant bit).

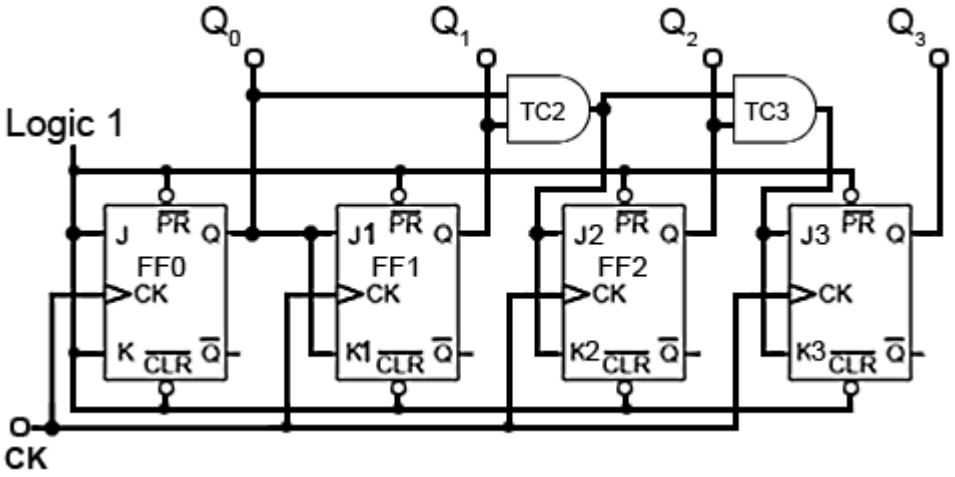


Fig. 5.6.8 Four Bit Synchronous Up Counter

Fig. 5.6.8 shows the additional gating for a four stage synchronous counter. Here FF3 is put into toggle mode by making J3 and K3 logic 1, only when Q_0 Q_1 and Q_2 are all at logic 1.

Q_3 therefore will not toggle to its high state until the eighth clock pulse, and will remain high until the sixteenth clock pulse. After this pulse, all the Q outputs will return to zero.

Note that for this basic form of the synchronous counter to work, the PR and CLR inputs must also be all at logic 1, (their inactive state) as shown in Fig. 5.6.8.

Table 5.6.2								
CK	Q_3	Q_2	Q_1	Q_0	$\overline{Q_3}$	$\overline{Q_2}$	$\overline{Q_1}$	$\overline{Q_0}$
0	0	0	0	0	1	1	1	1
1	0	0	0	1	1	1	1	0
2	0	0	1	0	1	1	0	1
3	0	0	1	1	1	1	0	0
4	0	1	0	0	1	0	1	1
5	0	1	0	1	1	0	1	0
6	0	1	1	0	1	0	0	1
7	0	1	1	1	1	0	0	0
8	1	0	0	0	0	1	1	1
9	1	0	0	1	0	1	1	0
10	1	0	1	0	0	1	0	1
11	1	0	1	1	0	1	0	0
12	1	1	0	0	0	0	1	1
13	1	1	0	1	0	0	1	0
14	1	1	1	0	0	0	0	1
15	1	1	1	1	0	0	0	0

Synchronous Down Counter

Converting the synchronous up counter to count down is simply a matter of reversing the count. If all of the ones and zeros in the 0 to 15_{10} sequence shown in Table 5.6.2 are complemented, (shown with a pink background) the sequence becomes $15_{10}$to 0.

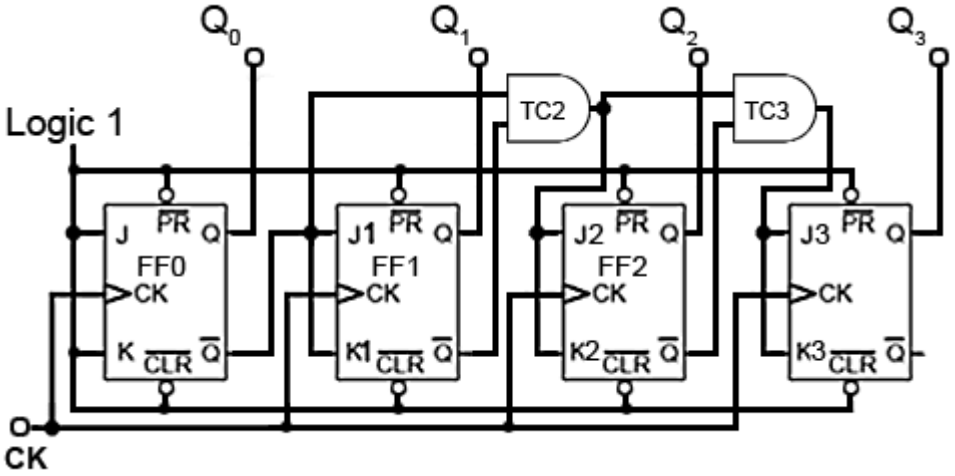


Fig. 5.6.9 Four Bit Synchronous Down Counter

Down Counter Circuit

As every Q output on the JK flip-flops has its complement on Q, all that is needed to convert the up counter in Fig. 5.6.8 to the down counter shown in Fig 5.6.9 is to take the JK inputs for FF1 from the Q output of FF0 instead of the Q output. Gate TC2 now takes its inputs from the Q outputs of FF0 and FF1, and TC3 also takes its input from FF2 Q output.

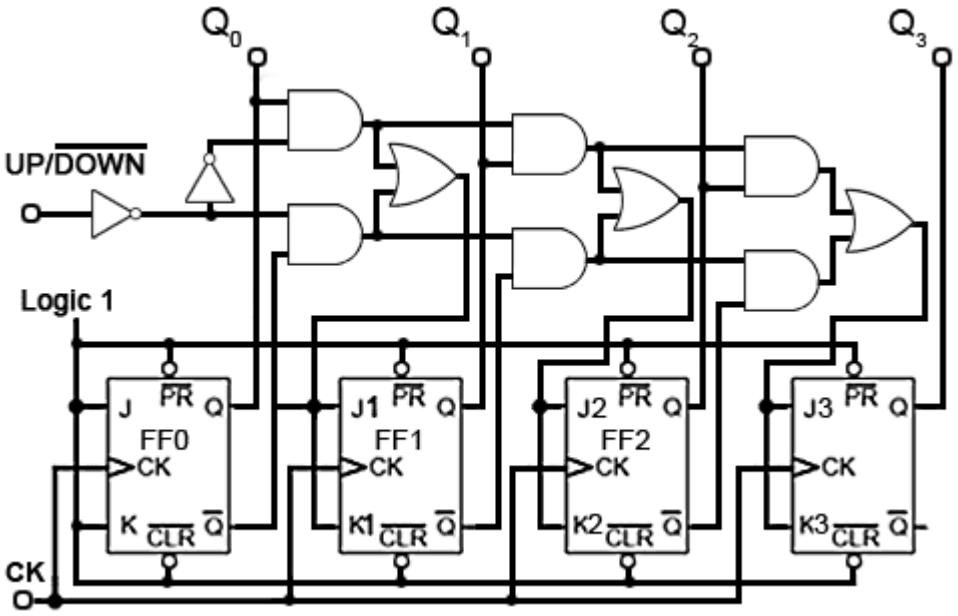


Fig. Fig. 5.6.10 Four-bit Synchronous Up/Down Counter

Up/Down Counter

Fig. 5.6.10 illustrates how a single input, called (UP/DOWN) can be used to make a single counter count either up or down, depending on the logic state at the UP/DOWN input.

Each group of gates between successive flip-flops is in fact a modified data select circuit described in Combinational Logic Module 4.2, but in this version an AND/OR combination is used in preference to its DeMorganequivalent NAND gate circuit. This is necessary to provide the correct logic state for the next data selector.

The Q and Q outputs of flip-flops FF0, FF1 and FF2 are connected to what are, in effect, the A and B data inputs of the data selectors. If the control input is at logic 1 then the CK pulse to the next flip-flop is fed from the Q output, making the counter an UP counter, but if the control input is 0 then CK pulses are fed from Q and the counter is a DOWN counter.

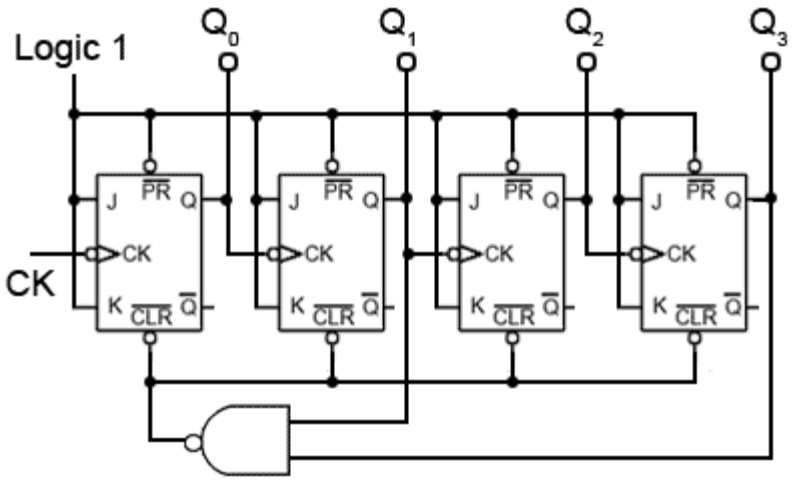


Fig. Fig. 5.6.11 Four-bit BCD Up Counter

Synchronous BCD Up Counter

A typical use of the CLR inputs is illustrated in the BCD Up Counter in Fig 5.6.11. The counter outputs Q_1 and Q_3 are connected to the inputs of a NAND gate, the output of which is taken to the CLR inputs of all four flip-flops. When Q_1 and $Q_3$are both at logic 1, the output terminal of the limit detection NAND gate (LD1) will become logic 0 and reset all the flip-flop outputs to logic 0.

Because the first time Q_1 and Q_3 are both at logic 1 during a 0 to 15_{10} count is at a count of ten (1010_2), this will cause the counter to count from 0 to 9_{10} and then reset to 0, omitting 10_{10} to 15_{10}.

The circuit is therefore a BCD_{8421} counter, an extremely useful device for driving numeric displays via a BCD to 7-segment decoder etc. However by re-designing the gating system to produce logic 0 at the CLR inputs for a different maximum value, any count other than 0 to 15 can be achieved.

If you already have a simulator such as Logisim installed on your computer, why not try designing an Octal up counter for example.

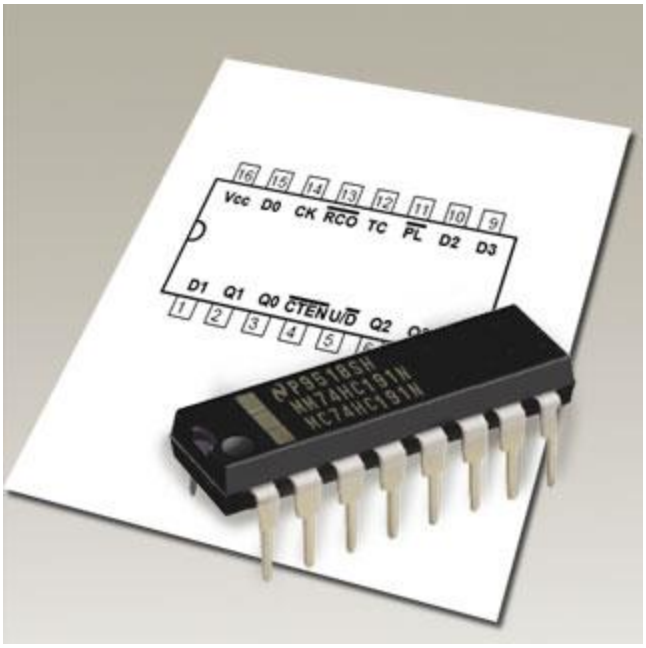


Fig. 5.6.12 Counter IC Inputs and Outputs

Counter IC Inputs and Outputs

Although synchronous counters can be, and are built from individual JK flip-flops, in many circuits they will be ether built into dedicated counter ICs, or into other large scale integrated circuits (LSICs).

For many applications the counters contained within ICs have extra inputs and outputs added to increase the counters versatility. The differences between many commercial counter ICs are basically the different input and output facilities offered. Some of which are described below. Notice that many of these inputs are active low; this derives from the fact that in earlier TTL devices any unconnected input would float up to logic 1 and hence become inactive. However leaving inputs un-connected is not good practice, especially CMOS inputs, which float between logic states, and could easily be activated to either valid logic state by random noise in the circuit, therefore ANY unused input should be permanently connected to its inactive logic state.

Enable Inputs

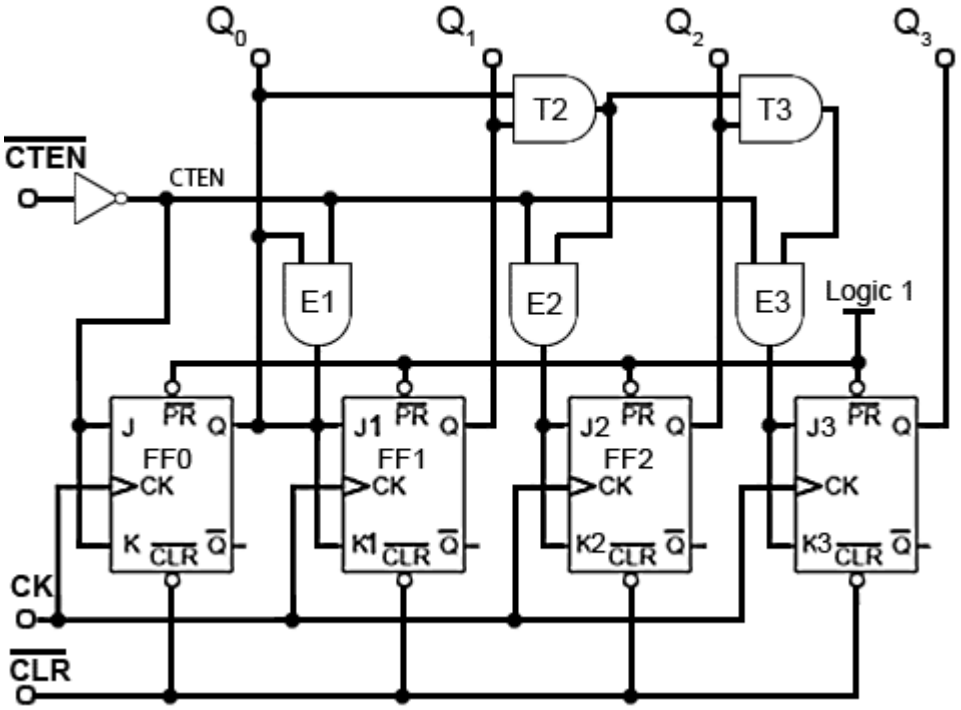


Fig. 5.6.13 Synchronous Up Counter with Count Enable and Clear Inputs

Logisim Simulation Available

ENABLE (EN) inputs on counter ICs may have a number of different names, e.g. Chip Enable (CE), Count Enable (CTEN), Output Enable (ON) etc., each denoting the same or similar functions.

Count Enable (CTEN) for example, is a feature on counter integrated circuits, and in the synchronous counter illustrated in Fig 5.6.13, is an active low input. When it is set to logic 1, it will prevent the count from progressing, even in the presence of clock pulses, but the count will continue normally when CTEN is at logic 0.

A common way of disabling the counter, whilst retaining any current data on the Q outputs, is to inhibit the toggle action of the JK flip-flops whilst CTEN is inactive (logic 1), by making the JK inputs of all the flip-flops logic 0. However, as the logic states of the JK inputs of FF1, FF2 and FF3 depend on the state of the previous Q output, either directly or via gates T2 and T3, in order to preserve the output data, the Q outputs must be isolated from the JK inputs whenever CTEN is logic 1, but the Q outputs must connect to the JK inputs when CTEN is at logic 0 (the count enabled state).

This is achieved by using the extra (AND) enable gates, E1, E2 and E3, each of which have one of their inputs connected to CTEN (the inverse of CTEN). When the count is disabled, CTEN and therefore one of the inputs on each of , E1, E2 and E3 will be at logic 0, which will cause these enable gate outputs, and the flip-flop JK inputs to also be at logic 0, whatever logic states are present on the Q outputs, and also

at the other enable gate inputs. Therefore whenever CTEN is at logic 1 the count is disabled.

When CTEN is at logic 0 however, CTEN will be logic 1 and E1, E2 and E3 will be enabled, causing whatever logic state is present on the Q outputs to be passed to the JK inputs. In this condition, when the next clock pulse is received at the CK input the flip-flops will toggle, following their normal sequence.

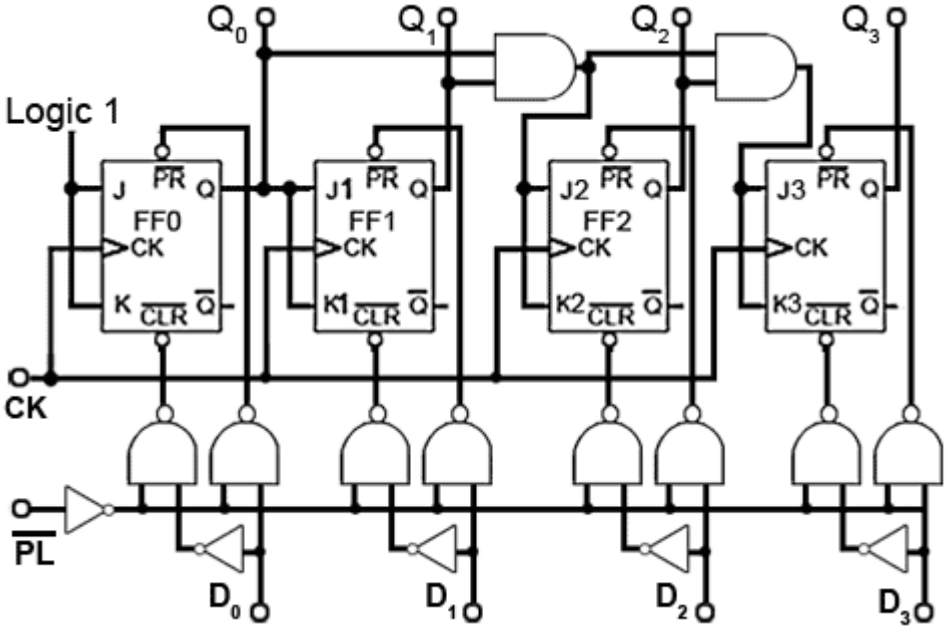


Fig. 5.6.14 Asynchronous Parallel Load

Logisim Simulation Available

Asynchronous Parallel Load

While common PR and CLR inputs can produce outputs of 0000 or 1111, a PARALLEL LOAD (PL) input will allow any value to be loaded into the counter. Using a separate DATA input for each flip-flop, and a small amount of extra logic, a logic 0 on the PL will load the counter with any pre-determined binary value before the start of, or during the count. A method of achieving asynchronous parallel loading on a synchronous counter is shown in Fig. 5.6.14.

Load Operation

The binary value to be loaded into the counter is applied to inputs D_0 to D_3 and a logic 0 pulse is applied to the PLinput. This logic 0 is inverted and applied to one input of each of the eight NAND gates to enable them. If the value to be loaded into a particular flip-flop is logic 1, this makes the inputs of the right hand NAND gate 1,1 and due to the inverter between the pair of NAND gates for that particular input, the left hand NAND gate inputs will be 1,0.

The result of this is that logic 0 is applied to the flip-flop PR input and logic 1 is applied to the CLR input. This combination sets the Q output to logic 1, the same value that was applied to the D input. Similarly if a D input is at logic 0 the output of the left hand NAND gate of the pair will be Logic 0 and the right hand gate output will be logic 1, which will clear the Q output of the flip-flop. Because the PL input is common to each pair of load NAND gates, all four flip-flops are loaded simultaneously with the value, either 1 or 0 present at its particular D input.

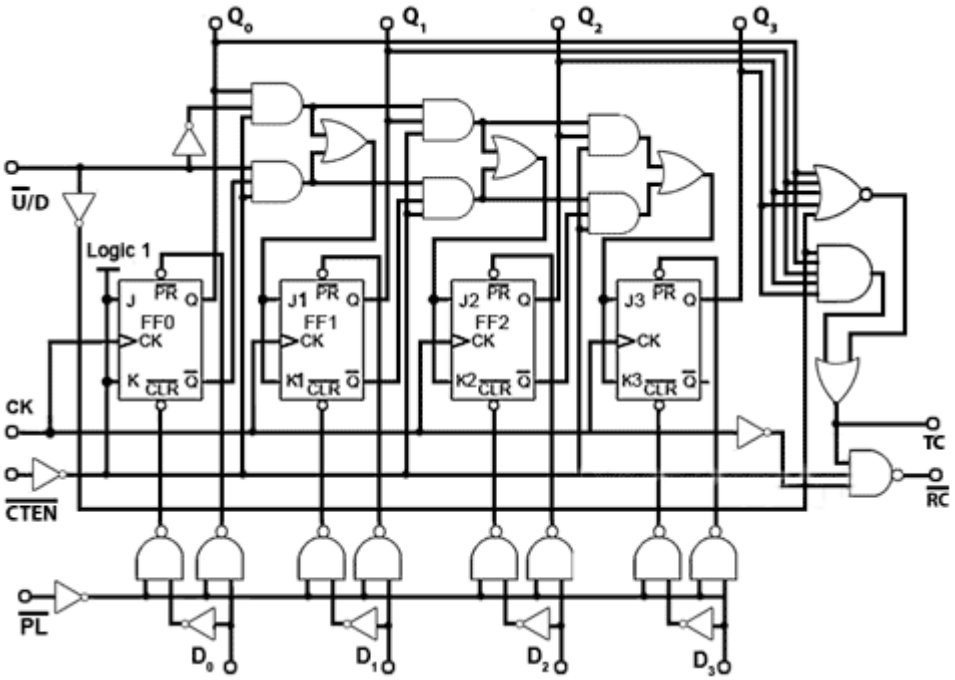


Fig. 5.6.15 Synchronous Up/Down Counter with Multiple Inputs and Outputs

Multiple Inputs and Outputs

Modifications such as those described in this module make the basic synchronous counter much more versatile. Both TTL and CMOS synchronous counters are available in the 74 series of ICs containing usually 4-bit counters with these and other modifications for a wide variety of applications. Fig 5.6.15 shows how all the input functions described above, plus some important outputs such as Ripple Carry (RC) and Terminal Count (TC) can be combined to form a single synchronous counter IC.

A typical single synchronous IC such as the 74HC191 four-bit binary up/down counter also uses these input and output functions, which are designated on NXP versions (Fig. 5.6.16) as follows:

Inputs

• D_0, D_1, D_2 and D_3 (Load inputs) - A 4 bit binary number may be loaded into the counter via these inputs when the Parallel Load input PL is at logic 0.

• CE (Count Enable) - Allows the count to proceed when at 0. Stops count without resetting when at logic 1.

• U/D (Up/Down) - Counts up when 0, down when at logic 1.

• CP - Clock Pulse input.

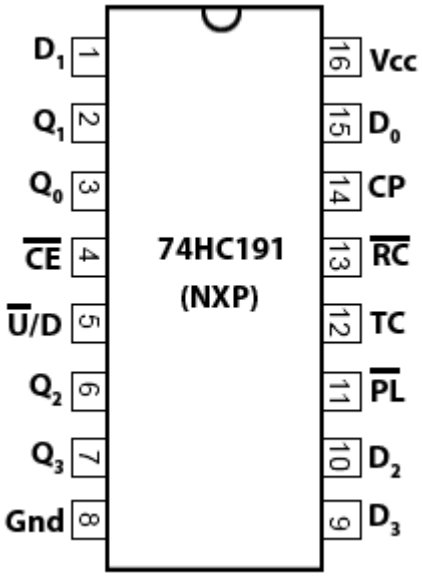


Fig. 5.6.16 74HC191 Pinout

Outputs

• Q_0, Q_1, Q_2 and Q_3 - Four bit binary output.

• TC (Terminal Count) - Also called MAX/MIN in some versions, gives a logic 1 pulse, equal in width to one full clock cycle, at each change over of the most significant bit (signifying that the count has overflowed beyond the end of an up or down count). TC can be used to detect the end of an up or down count, and as well as being available as an output, TC is used internally to generate the Ripple Carry output.

• RC (Ripple Carry) - Outputs a logic 0 pulse, equal in width to the low portion of the clock cycle at the end of a count, and when connected to the clock input of another 74HC191 IC it acts as a ‘carry’ to the next counter.

Cascading Synchronous Counters

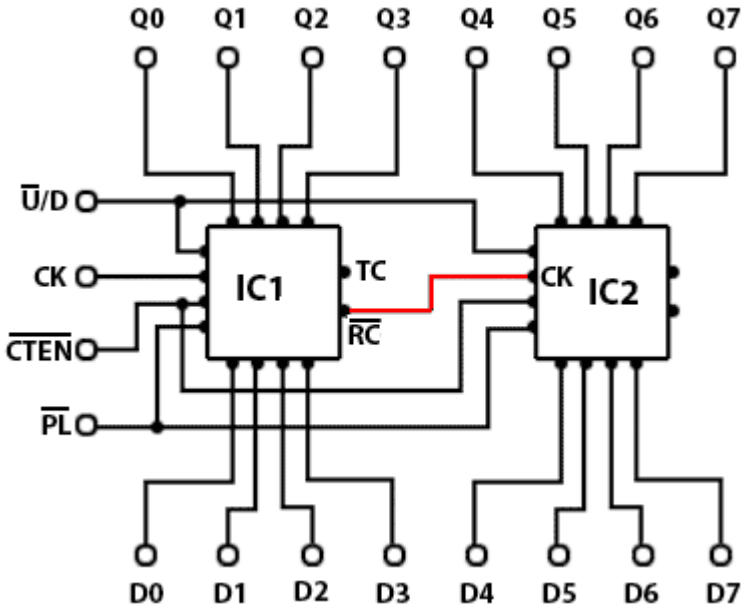


Fig. 5.6.17 Connecting the 74HC191 in Cascade

Connecting Synchronous counters in cascade, to obtain greater count ranges, is made simple in ICs such as the 74HC191 by using the ripple carry (RC) output of the IC counting the least significant 4 bits, to drive the clock input of the next most significant IC, as show in red in Fig. 5.6.17.

Although it may appear that either the TC or the RC outputs could drive the next clock input, the TC output is not intended for this purpose, as timing issues can occur.

Synchronous vs. Asynchronous Counters

Although synchronous counters have a great advantage over asynchronous or ripple counters in regard to reducing timing problems, there are situations where ripple counters have an advantage over synchronous counters.

When used at high speeds, only the first flip-flop in the ripple counter chain runs at the clock frequency. Each subsequent flip-flop runs at half the frequency of the previous one. In synchronous counters, with every stage operating at very high clock frequencies, stray capacitive coupling between the counter and other components and within the counter itself is more likely occur, so that in synchronous counters interference can be transferred between different stages of the counter, upsetting the count if adequate decoupling is not provided. This problem is reduced in ripple counters due to the lower frequencies in most of the stages.

Also, because the clock pulses applied to synchronous counters must charge, and discharge the input capacitance of every flip-flop simultaneously; synchronous counters having many flip-flops will cause large pulses of charge and discharge

current in the clock driver circuits every time the clock changes logic state. This can also cause unwelcome spikes on the supply lines that could cause problems elsewhere in the digital circuitry. This is less of a problem with asynchronous counters, as the clock is only driving the first flip-flop in the counter chain.

Asynchronous counters are mostly used for frequency division applications and for generating time delays. In either of these applications the timing of individual outputs is not likely to cause a problem to external circuitry, and the fact that most of the stages in the counter run at much lower frequencies than the input clock, greatly reduces any problem of high frequency noise interference to surrounding components.