Chapter 7: Systems for Frequency Generation

Chapter 7 Objectives

At the conclusion of this chapter, the reader will be able to:

- List the components in a Phase Locked Loop, explaining the purpose of each one.
- Describe the three operating states of a PLL.
- Given the parameters of the loop, calculate the frequencies in each part of a PLL.
- Draw a block diagram of a direct PLL frequency synthesizer.
- Calculate the frequencies and divisors needed in a direct PLL synthesizer.
- Calculate the frequencies in the loop of an indirect PLL synthesizer
- Describe the software events necessary for control of a PLL synthesizer.
- Draw a block diagram of a DDS frequency synthesizer
- Calculate the various parameters for a DDS frequency synthesizer.
- Given block and schematic diagrams of a frequency synthesizer, develop a plan for troubleshooting it.

All radio transmitters and receivers use *oscillators* to provide needed frequencies. An oscillator is a stage that converts DC from the power supply into an AC output signal at a specified frequency. Up to this point we have studied two ways of controlling the frequency of an oscillator. These two methods are *discrete LC control* and *crystal control*.

An oscillator's frequency can be controlled by a discrete LC tank. This approach is simple, and the oscillator's frequency is easily adjusted by varying either the L or C component values within the tank. However, this approach doesn't provide a very stable output frequency; the Q of the LC tank is too low to keep the frequency steady.

Crystal control of an oscillator's frequency provides rock-stable output. This would be the ultimate choice for all transmitters and receivers, except that the frequency of a crystal oscillator cannot be appreciably changed without replacing the crystal. When many different operating frequencies are required, this approach becomes very expensive.

A *frequency synthesizer* is a circuit that *synthesizes* or "builds" new frequencies. These new frequencies are based on a highly stable frequency source, which is usually a single quartz crystal oscillator. The stable frequency source is called the *reference* or *master* oscillator.

Modern frequency synthesizers are digitally controlled. They make possible all sorts of products and applications, from electronically tuned shirt pocket stereo receivers, to sophisticated commercial communication and navigation equipment. Digital control of frequency allows microprocessor control of radio features. The frequency synthesizer in a typical product is merely an input/output (I/O) device connected to the controlling CPU. Software calls the shots, and analog hardware does the work.

The mix of analog and digital hardware in a frequency synthesizer can be very intimidating to the technician, especially when there is a hidden piece of computer software running the show. No matter how complex, all frequency synthesizers are based on a few basic ideas. By learning these principles, you'll be well prepared for working with and troubleshooting these fundamental communications building blocks.

7-1 The Phase Locked Loop

The phase-locked loop, or PLL, is one of the most useful blocks in modern electronic circuits. It is used in many different applications, ranging from communications (FM modulation, demodulation, frequency synthesis, signal correlation), control systems (motor control, tracking controls, and so on), as well as applications such as pulse recovery and frequency multiplication. Knowing PLLs can really boost your "tech IQ!"

PLL Theory of Operation

A PLL is a *closed-loop* system, whose purpose is to lock its oscillator onto a provided input frequency (sometimes called the *reference* frequency.) A closed-loop system has feedback from output to input. In a PLL the feedback is negative, meaning that the system is *self-correcting*. When we say that the PLL's oscillator is *locked* onto the reference, we mean that there is *zero frequency difference (error) between the PLL oscillator and the reference frequency*. It might not make sense at this point as to why we would want to "lock" one oscillator onto another's frequency. Can't we just take the output from the reference oscillator and be done?

There are two reasons why we will want to do exactly this. First, the PLL provides *filtering* action. A PLL can lock onto a noisy reference signal, providing a filtered output that is relatively free of noise. Second, by modifying the PLL feedback loop we can derive new frequencies from the reference signal. We can build a

"tunable" frequency source based on a rock-solid crystal frequency reference. This is *frequency synthesis*. Figure 7-1 shows the basic elements in a PLL.

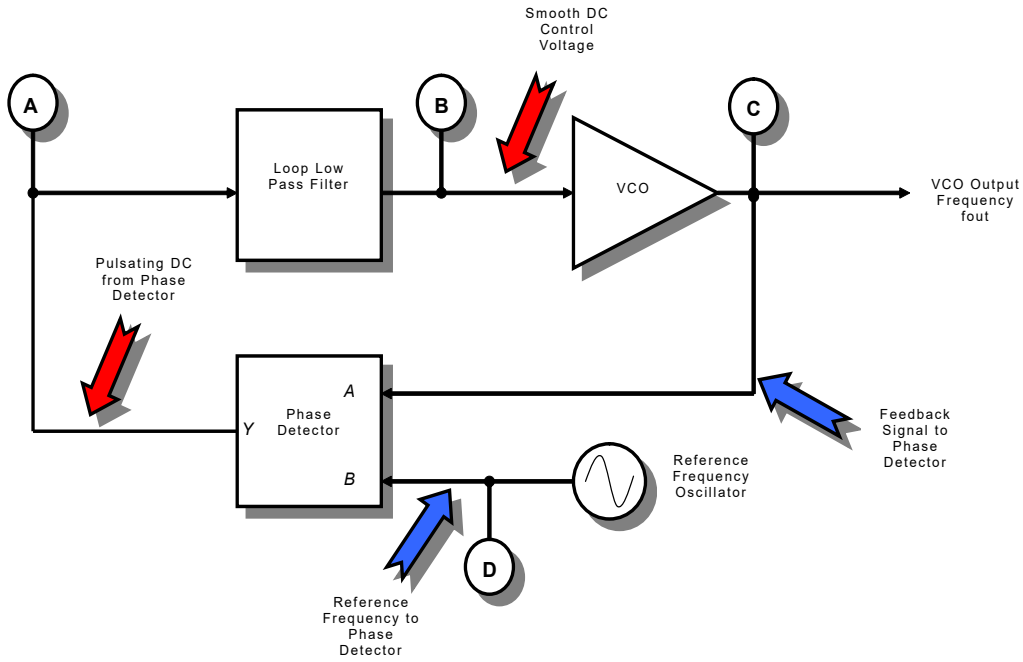

Figure 7-1: Phase Locked Loop Block Diagram

A PLL consists of a *voltage controlled oscillator*, or *VCO*, a *phase detector*, and a *loop filter*. Each of these components has a special job in keeping the PLL locked onto the reference frequency.

Voltage Controlled Oscillator

A PLL has a special oscillator, a VCO. Previous oscillators we have studied depended on either an LC resonant circuit or a crystal to determine their frequency. The output frequency of a VCO depends on an LC or RC circuit, and a *control voltage*. The LC or RC portion of the circuit determines the approximate frequency range that the VCO will operate in, and the control voltage moves the VCO frequency up or down within that range.

Most technicians think of a VCO as a *voltage to frequency converter*, since the input to a VCO is a DC control voltage, and the output of a VCO is a varying frequency. Figure 7-2 is a graph of the *transfer characteristic* of a simple VCO circuit. The transfer characteristic is the input-output relationship.

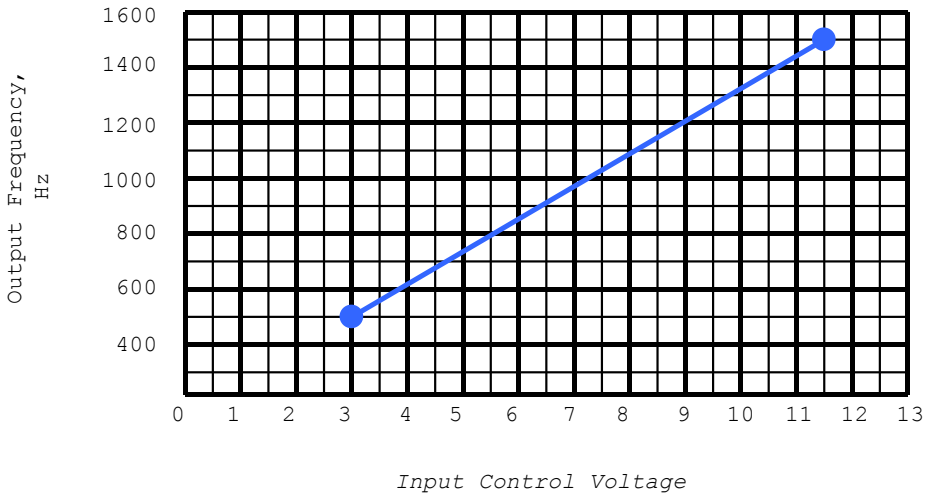

Figure 7-2: Transfer Characteristic of a VCO

Note the units on the axes in Figure 7-2. The horizontal axis has units of voltage, and the vertical shows frequency. This shows us that the output frequency of the device depends upon the input control voltage.

There are definite limits on how high and how low the frequency of the VCO can go. These limits are inherent to any VCO, and are determined by the circuit designers. Most practical VCO circuits do not operate over more than about a 3:1 frequency range.

Example 7-1

What will the output frequency of the VCO of Figure 7-2 be if the control voltage is (a) 3 V ; (b) 8 V ; (c) 12 V

Solution
The output frequencies can be read directly from the graph of Figure 7-2:
When Vc = 3 V, f_{out} = 500 Hz
When Vc = 8 V, f_{out} = 1100 Hz
When Vc = 12 V, f_{out} = undefined. The VCO isn't designed to accept a control voltage above 11.5 volts. We know this because the graph stops at Vc = 11.5 V.

Phase Detector

The purpose of a PLL is to lock the VCO frequency onto the reference frequency. In order for this to happen, a decision must be made about the VCO's frequency: Is the VCO frequency too high, too low, or just right? A special section of the PLL, the *phase detector*, makes this decision.

The decision making process uses feedback. It's very much like the cruise control in a car. Suppose that you have set the cruise speed in your car to be 70 MPH. A sensing device measures the car's speed. That speed information is sent to the cruise control unit. *The speed information is the feedback*. If the car is moving too slowly the cruise control will respond by opening the throttle a little wider, which will bring the vehicle speed up to the desired point. The opposite will happen if the car is moving too quickly; the throttle is closed to slow down. The speed of the car is not expected to always be *exactly* the same as the set point of the cruise control. There is always a small error allowed. This is necessary to prevent stability problems (such as "hunting" -- where the control system never settles to a steady-state condition).

In a PLL, the VCO control voltage is like the throttle in a car, and the resulting VCO frequency is analogous to the car's speed. The *phase detector* compares the VCO frequency to that of the reference. In Figure 7-1 you can see that the VCO output signal is fed back into the A input of the phase detector. The B input of the phase detector sees the reference signal. Unlike the cruise control in a car, the phase detector decision-maker will not be satisfied until there is *zero frequency difference (error)* between the VCO and the reference source. In fact, this can be stated as a simple law[4].

Finley's Law

Finley's Law for Phase Detectors

If the two inputs of a phase detector are not at exactly the same frequency, then the phase detector output will be in either positive or negative saturation.

How can the phase detector possibly achieve *zero* error in frequency? The answer becomes evident when we look at the relationship between frequency and phase. Suppose that we stretch the car analogy a little further by imagining two cars traveling in the same direction down a four-lane highway. The "frequency" of each car is indicated by its speedometer. The "phase" of the cars is just their relative position on the highway. In Figure 7-3, although both cars are not perfectly in phase, they are both moving at exactly the same speed. Their "frequencies" are identical. The cars stay together as they travel down the highway. *The phase of two signals does not have to be the same for their frequencies to be identical. However, the phase error must be constant.*

[4] Finley's Law is a special case of Mason's Rule, a formulation for determining the input-output relationship for closed-loop systems. With Finley's Law, we allow the forward gain of a system to approach infinity. In more advanced coursework, you'll learn that such systems have zero steady-state error thanks to the use of integral control methods. As a practitioner, simply remember that a properly-operating PLL always has zero frequency error.

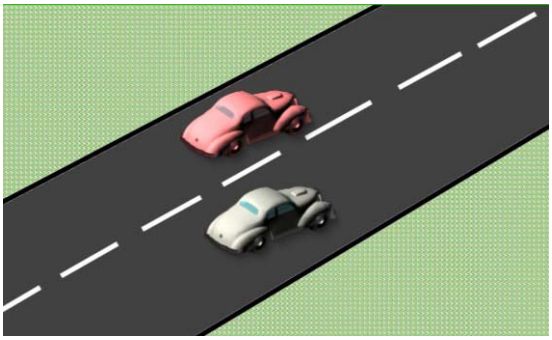

Figure 7-3: The Frequencies are Identical, the Phases are not. The Phase Error is Static and the Cars Stay in the Same Relative Positions as they Travel.

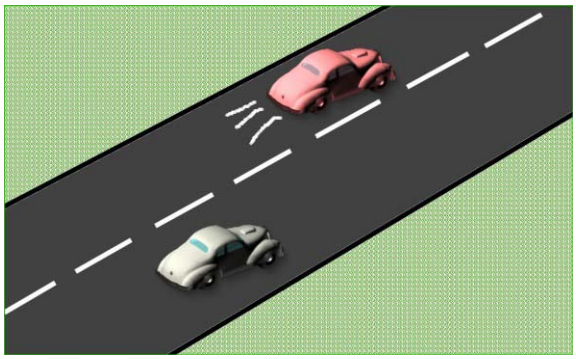

Figure 7-4: The Phase Error is no longer Constant; the Frequencies (Speeds) are not Exactly Equal.

In Figure 7-4, the left car has sped up. The two cars are no longer in a fixed relationship with each other. There is now an *increasing* distance (phase difference) between the cars. Their speeds (frequencies) are no longer equal.

A Locked PLL has Zero Frequency Error

A phase detector achieves zero frequency error by comparing phase. When the phase difference between two signals is constant, their frequencies are identical. This is why Finley's Law is true for phase detectors. This special property makes the phase detector an excellent frequency "referee" for the PLL. It also explains why there is always zero frequency error in a PLL once it is in lock.

Types of Phase Detectors

Analog Phase Detectors

It is possible to build both analog and digital phase detectors. An analog phase detector is nothing more than our old friend, the mixer. You might recall that given two frequency inputs (*f1* and *f2*), the outputs of an ideal mixer are the original frequencies (*f1* and *f2*), the sum of the two frequencies (*f1* + *f2*) and the difference (*f1* - *f2*).

If the two frequencies *f1* and *f2* are exactly equal, their difference is zero Hz. That's DC! With trigonometry, we can demonstrate that the DC output of a mixer, given identical input frequencies *f1* and *f2*, is proportional to the phase difference between the two inputs.

In an analog phase detector, we recover the DC level with a low-pass filter at the detector's output.

Digital Phase Detectors

Most phase detectors are digital. The reason for this is quite simple: All the components of a digital phase detector can be constructed on a single IC chip. This greatly decreases cost and improves reliability.

Digital phase detectors are built using digital logic gates; therefore, the output of a digital phase detector is pulsating DC with a varying duty cycle. One of the simplest possible phase detectors is an exclusive-OR logic (XOR) gate, as shown in Figure 7-5(a).

XOR Phase Detector Waveforms

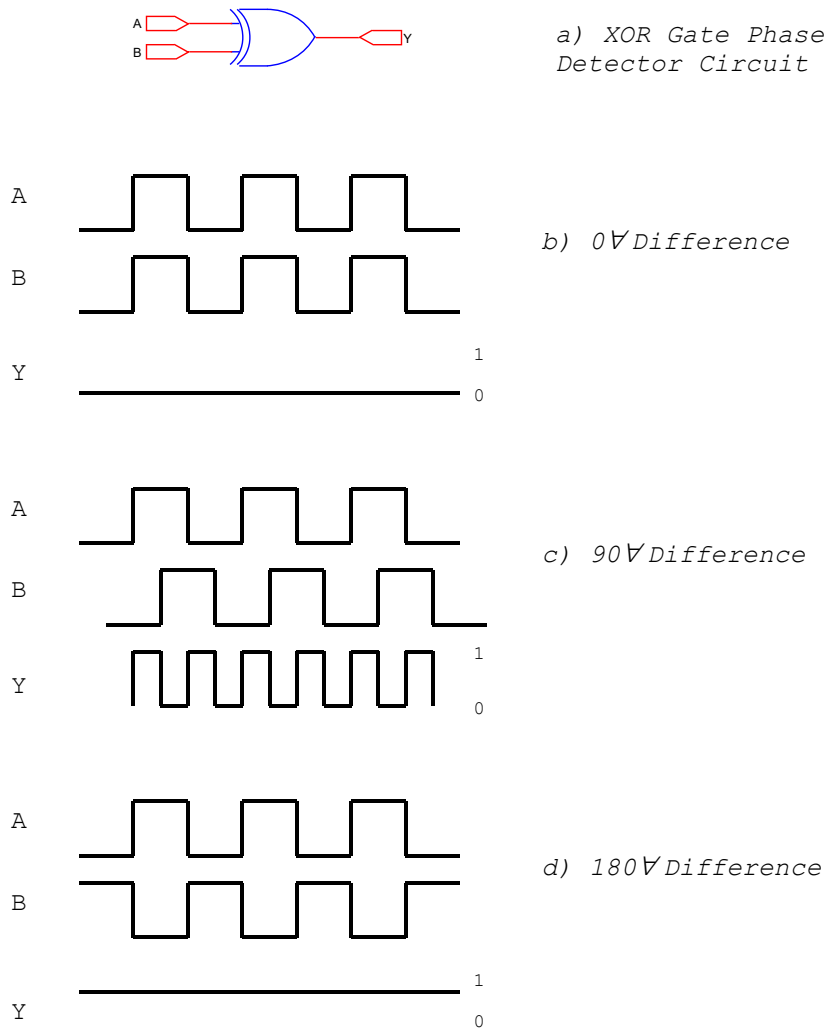

Figure 7-5: XOR-type phase detector and circuit waveforms

An exclusive-OR gate gives a "1" output whenever the inputs are opposites, and a "0" output when the inputs are the same. In Figure 7-5(b), the two inputs are precisely in phase, which means they're the same all the time. The gate always produces a "0" output, which corresponds to 0 volts.

In Figure 7-5(c), the "B" input is *leading* the "A" input by 90∀ Now the gate inputs are different at parts of the cycle, and consequently, the output "Y" goes high precisely one-half of the time. We would say that its duty cycle is 50%, and that its average voltage is $V_{cc}/2$.

As we increase the phase difference to 180∀, the output stays high all the time. The duty cycle is now 100%, and the average voltage is V_{cc}. The XOR gate has converted the input phase difference into an average DC voltage. This DC voltage is pulsating at twice the frequency of the input signals. Figure 7-6 illustrates the transfer characteristic of this phase detector.

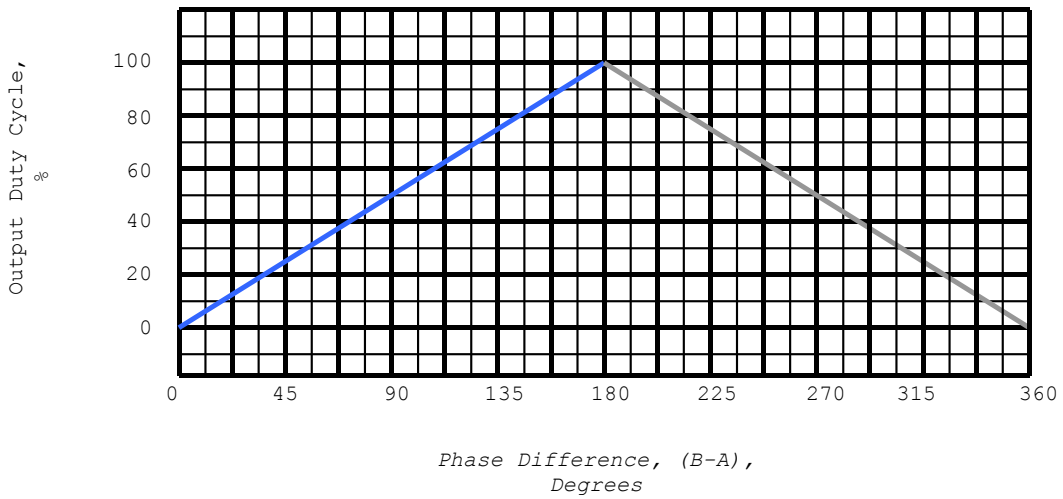

Figure 7-6: Transfer Characteristic of the Exclusive-OR Phase Detector

XOR Phase Detector Limitations

The blue graph in Figure 7-6 reflects operation from a 0 to 180∀difference. As the phase difference increases, the average output voltage increases. But something strange happens as the phase difference passes 180∀(gray region of graph) -- the output voltage starts *decreasing*! The XOR phase detector simply cannot operate over more than a 180∀range.

The XOR phase detector has another problem. It is sensitive to the duty cycle of the two input signals, which are required to be square waves. If either signal varies in duty cycle, the output will falsely indicate a phase change.

In order to overcome this limitation, more complex logic circuits are used in actual PLL chips. These circuits are typically sensitive to the edges of the waveforms -- they are "clocked" detectors. A practical phase detector usually includes a Schmitt trigger on each input in order to convert the input signals to clean square waves. Edge-sensitive (clocked) phase detectors are not sensitive to the duty cycle of the input signals. Chips such as the Motorola CD4046 include both XOR and clocked phase detectors.

Loop Filter

The "Y" output of the phase detector in the PLL of Figure 7-1 is a pulsating DC voltage with a varying duty cycle. The bigger the phase difference becomes (within certain limits, of course), the larger this duty cycle becomes, and the larger the average voltage being fed back into the VCO on top. This voltage will tend to correct the frequency of the VCO, either raising or lowering its frequency. But there's a problem here.

The VCO needs a nice, steady DC voltage at its control voltage input. Can you imagine the effect of the pulsating DC on the VCO? Think of a car that only has two throttle positions, wide open and off. The desired speed is 70 MPH. We're traveling 69 MPH, which is too slow -- so we must choose the *wide open* throttle position. The car lurches forward with this throttle application, overshooting the target speed of 70 MPH. Now our only choice is to totally close the throttle and jam on the brakes. The passengers are thrown forward as the car rapidly decelerates. The cycle repeats, over and over. The motion of the car isn't very smooth at all, although its average speed is very close to 70 MPH. (This by the way is "hunting" behavior. The system can't settle on a stable output.)

The same thing will happen with the VCO frequency without careful filtration of the phase detector's output. We need its output frequency to be steady, like the reference input. To accomplish, we smooth out the pulsating DC from the phase detector into a steady *DC average* voltage. This is the purpose of the *loop filter*. This filter in effect smoothes the rough phase detector output waveform into a steady DC voltage for the VCO. The VCO will then be able to smoothly track the input reference frequency.

The loop filter in a PLL is usually a *low-pass* type. It can be a simple RC time constant, or something more involved. The RC time constant within the loop filter determines several of the loop's characteristics, including how fast it can respond to changes. A long loop filter RC time-constant provides excellent filtering but very slow response. By reducing the RC time constant, we can speed up the ability of the PLL to respond to changes -- but at the expense of poorer VCO control voltage filtering, which will show up as "jitter" (time domain) or "spurious sidebands" (frequency domain) in the VCO frequency output.

PLL Operating States

A PLL has three operating states. These are the *free-run*, *capture*, and *locked* conditions.

In the *free-run* state, there is no reference input frequency being provided to the PLL. Under this condition, there is nothing for the phase detector to compare the VCO output frequency with. The VCO "free runs" at its own natural frequency. The free-running frequency of the loop is normally determined by an LC or RC circuit within the VCO.

Applying a frequency to the *reference* input of the PLL causes the loop to go into the *capture* state. The capture condition normally doesn't last very long, for the PLL immediately tries to get locked onto the input frequency. The time needed to get locked depends partially on the RC time-constant to the loop filter, and the difference in frequency between the VCO and the applied reference input signal. Capture is very similar to the slipping of the clutch that takes place when a manual-transmission car takes off from a stop. Initially, there is a great difference between the input and output of the clutch; as the clutch pedal is released, the car gains speed (VCO moves toward reference frequency), the input and output of the clutch eventually become *exactly* equal -- the clutch is no longer slipping. When the VCO frequency exactly equals the reference frequency, the PLL has attained *lock*.

The PLL cannot lock onto all frequencies; only a certain range of frequencies within the *capture range* can be acquired if the PLL is initially in the free-running state. Usually, the free-running frequency is in the middle of the capture range. The width of the capture range is determined by PLL design; the loop low-pass filter is important in determining this.

The last PLL state is the desired state: *locked!* In this state, the PLL has successfully passed through the capture phase, and it has its VCO locked onto the input reference frequency. The PLL cannot remain locked for all frequencies, and if the input reference frequency moves outside the *lock range*, (which is usually larger than the capture range), the PLL will drop out of lock. The VCO is the primary component in the PLL that determines lock range, for the lock range is actually just lower and upper limits of VCO oscillation frequency.

Figure 7-7 illustrates the relationship between free-running frequency, capture range, and lock range for a typical PLL.

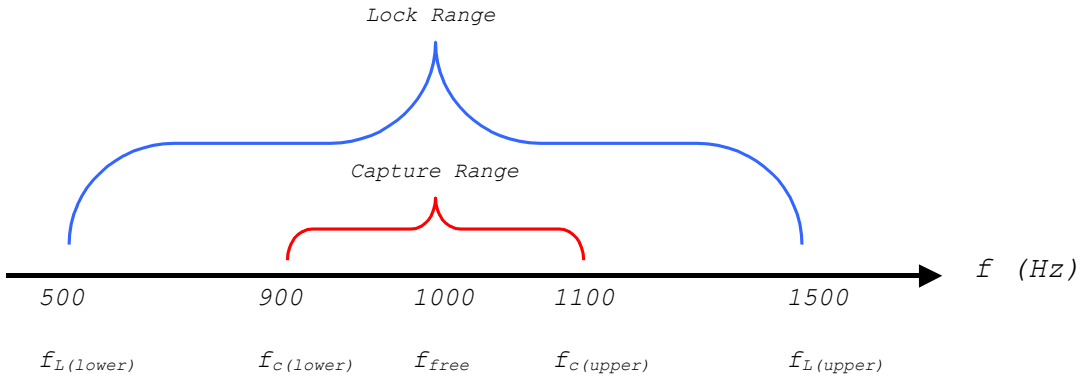

Figure 7-7: PLL operation regions

Example 7-2

Suppose that the PLL of Figure 7-1 has the operation regions given in Figure 7-7. Give the PLL state and VCO output frequency for each of the cases below. Assume that initially, there is *no* reference input frequency applied, and that the conditions attained in each case will apply to the next case.

a) $f_{reference}$ = 450 Hz b) $f_{reference}$ = 800 Hz c) $f_{reference}$ = 950 Hz d) $f_{reference}$ = 1450 Hz
e) $f_{reference}$ = 1550 Hz f) $f_{reference}$ = 1200 Hz g) $f_{reference}$ = 0 Hz

Solution

a) The loop is in the *capture* state, and the VCO frequency cannot be determined. The VCO is rapidly hunting up and down in frequency trying to match the reference, but since the frequency is too low (less than $f_{c(lower)}$), the loop can not acquire lock.

b) The loop is *still* in capture, and again, the VCO frequency is pretty much unknown.

c) The loop is in *lock*. The applied frequency has fallen inside the *capture range* (900 - 1100 Hz), so the VCO can "catch up" with the reference signal. *Finley's Law* applies when the loop is in lock, so $f_{VCO} = f_{reference}$ = 950 Hz.

d) Once the loop is in lock, the VCO can now follow the reference anywhere within the *lock range*. Since we attained lock already, the loop follows, and we get $f_{VCO} = f_{reference}$ = 1450 Hz.

e) The frequency 1550 Hz is outside of the lock range -- the VCO can't go that high. The loop drops out of lock, back into *capture* (since there is an applied reference signal). The VCO frequency is unknown.

f) This is weird, but true! Starting *unlocked* from condition (e), the loop will still be out of lock here. In order to gain lock, the frequency must first "dip" into the capture range. Therefore, the loop is in *capture* and the VCO frequency is still unknown.

g) The reference signal has been removed, and the loop free-runs again. The VCO frequency will be *approximately* 1000 Hz. Since the VCO frequency is determined by an LC or RC circuit, this frequency is not very accurate. The loop is in the *free-run* state again.

Determining Loop State with Instruments

A technician often needs to find out whether or not a PLL is properly locked. The most common method of doing this involves the use a dual-trace oscilloscope. Channel 1 of the scope is connected to the *reference* input of the phase detector (point D in Figure 7-1), and channel 2 is connected to the remaining phase detector input (point C). In Figure 7-1, this is exactly the same as the VCO output -- but this is not true for all PLLs.

The oscilloscope is usually set to trigger off the reference, and the resulting two waveforms (reference and VCO) are compared. If they're exactly at the same frequency, the loop is in lock. On a scope, this is immediately apparent as shown in Figure 7-8.

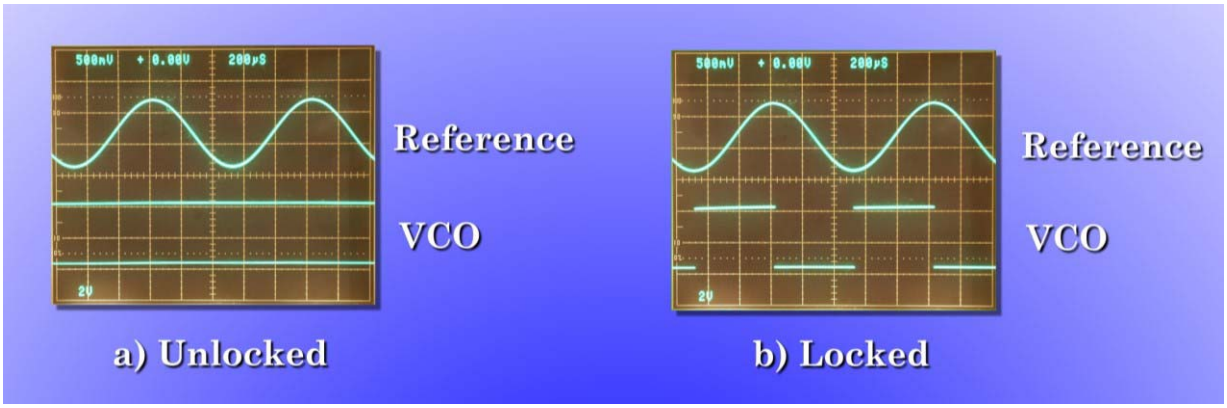

Figure 7-8: Assessing the PLL state with an oscilloscope

In Figure 7-8(a) the reference signal is stable since the scope is set to trigger from it. However, the VCO looks very strange. In this photo, the VCO appears as two lines. Actually, the VCO signal looked like it was "running" left to right on the display. The action of the camera blurred this into the two lines you see. *If the VCO output cannot be easily seen, the loop is definitely not in lock!*

Figure 7-8(b) shows the loop in lock. The VCO display does not appear to move or "crawl" across the screen. It has exactly the same frequency as the reference. You can verify this; both the reference and VCO have exactly the same *period* in Figure 7-8(b).

The loop state can also be verified by using a frequency counter. You've probably already guessed the two points of measurement -- that's right, each phase detector input. They must read *exactly* the same frequency, in a stable manner.

The scope method is very popular. It gives us instant feedback about the status of the loop. It's also better than attempting to read the two frequencies with a frequency counter. Some frequency counters have difficulty in properly triggering off an analog RF signal, and when they do trigger, the readout process is relatively slow (seconds versus a few ms for the scope update). Some frequency counters may excessively load an RF circuit, leading to false readings.

Section Checkpoint

7-1 Why are frequency synthesizers needed in modern electronics?
7-2 What is meant by the term "closed-loop system?"
7-3 List the parts of a PLL, explaining what each one does.
7-4 What is the primary action or goal of a PLL?
7-5 The VCO in a PLL converts _____ into _____.
7-6 Which part of a PLL acts as a decision maker?
7-7 Why is a low-pass filter necessary in a PLL?
7-8 State Finley's Law for Phase Detectors.
7-9 How much frequency error is present in a *locked* PLL?
7-10 What are the three PLL operating states?
7-11 What is the difference between the *capture* and *lock* ranges of a PLL?
7-12 Explain how to determine whether or not a PLL is in lock by using benchtop instruments.

7-2 PLL Synthesizers

The basic PLL configuration (Figure 7-1) itself synthesizes nothing; the VCO frequency of the loop is always equal to the reference input (when the system is in lock). By modifying the feedback portion of the loop, we can get the loop to produce new frequencies. In other words, we can convert the PLL frequency "follower" to a frequency *synthesizer* by altering the feedback sent back to the phase detector.

Frequency Dividers

A *frequency divider* is a circuit that divides an incoming frequency by some chosen number. Frequency dividers are really nothing more than digital counters. Figure 7-9 shows a *divide by two* circuit with waveforms:

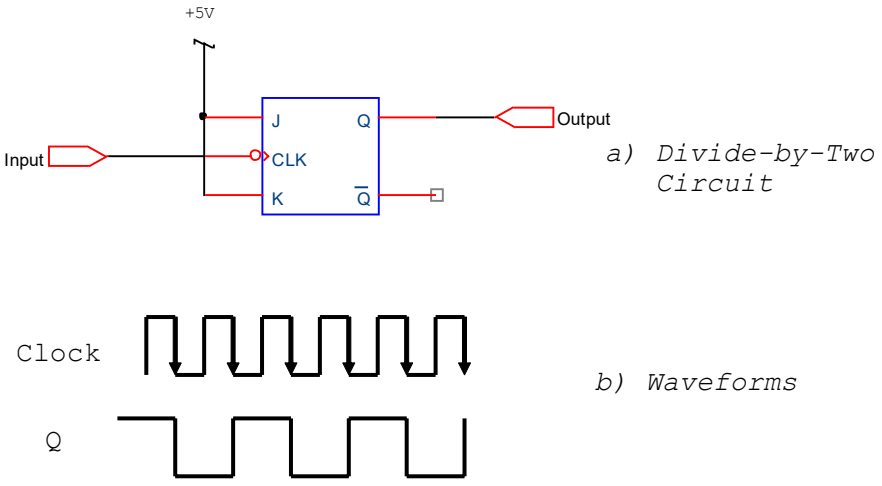

Figure 7-9: A Divide by Two Circuit

In Figure 7-9(a) a JK flip-flop is connected in the *toggle* mode. Recall that when both the J and K inputs are tied high, a JK flip-flop is "programmed" to toggle. The flip-flop will change state (toggle) on each active clock transition. For the circuit in the figure, we know that the JK's clock input is sensitive to the *falling edge* of the clock signal. Therefore, each time the clock input goes from high to low, the Q and /Q outputs of the flip-flop will change state.

Look at the relative frequencies of the *clock* and Q signals in Figure 7-9(b). There are *two* complete clock cycles for every Q cycle. Therefore, we can say that the frequency of Q is precisely 1/2 of the clock. *The circuit has divided the input clock frequency by two.* Another way of saying this is that the divider has a modulus of two.

Higher Divisor Ratios

By cascading counters, we can get larger divisors. Can you determine the divisor ratio for each of the circuits of Figure 7-10?

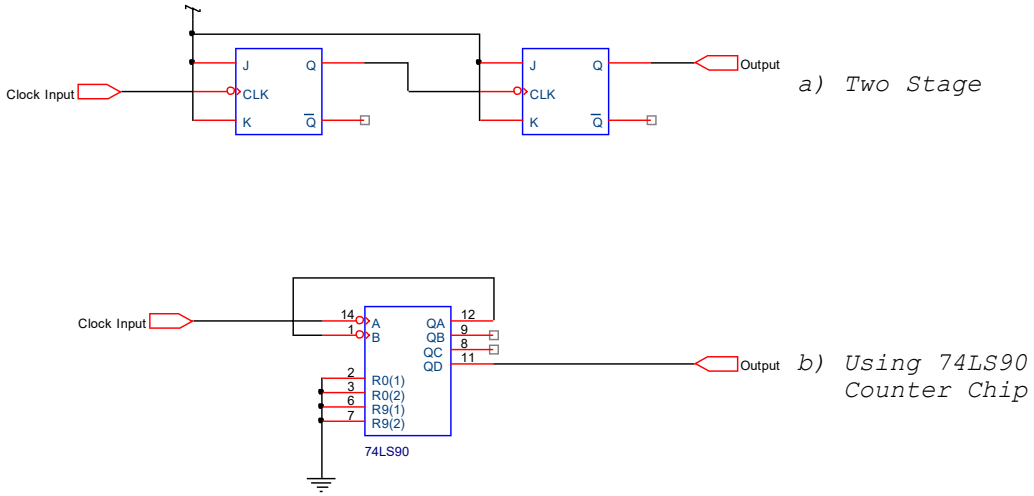

Figure 7-10: Two More Divider Circuits

In Figure 7-10(a) we have cascaded two JK flip-flops to form a *ripple* counter. From digital fundamentals, you'll recall that the modulus of a binary ripple counter is equal to:

(7-1) $MOD = 2^N$

Where N is the number of flip-flops in the circuit and *MOD* is the modulus, or number of unique counting states. Since $N = 2$, the circuit of Figure 7-10(a) divides by 2^2, or 4. The top circuit therefore divides the incoming frequency by four.

The bottom circuit cannot be analyzed without studying the data sheet for the 74LS90. The 74LS90 is wired as a BCD (modulo 10) counter in Figure 7-10(b). This means that there are ten unique counting states, and since the Q_D output is being used as the output, there will be one pulse on Q_D for every ten input pulses on the clock input (*A*). The circuit divides the input frequency by 10.

Frequency Dividers within a Loop

Figure 7-11 shows a PLL with a frequency divider inserted within the feedback loop. The addition of the divider within the feedback portion of the loop changes the signal that the phase detector (the loop's decision maker) sees.

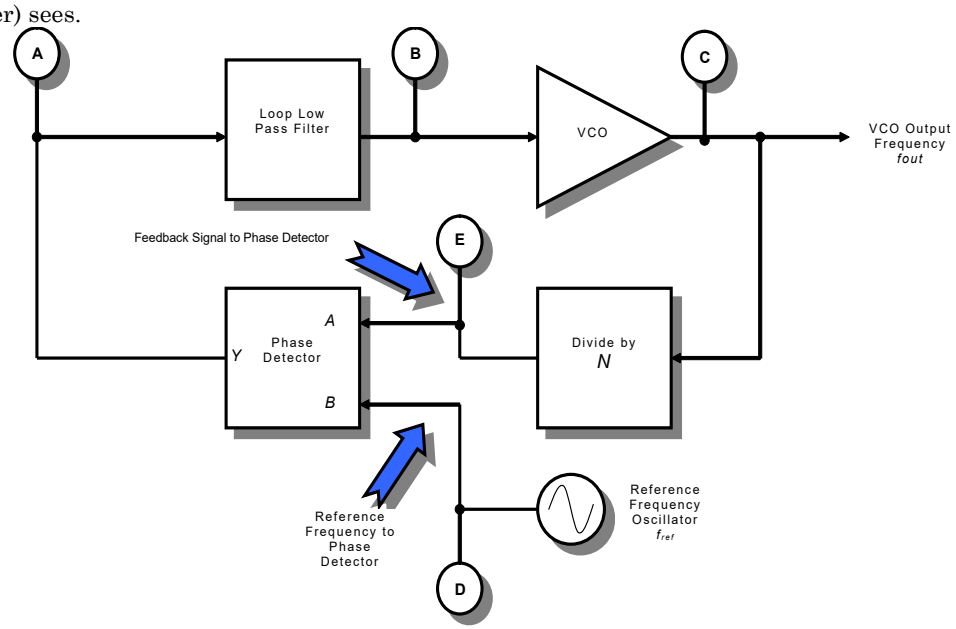

Figure 7-11: A PLL with a divider in the loop

To understand what will happen in the circuit of Figure 7-11, it is helpful to keep Finley's law in mind. This law states: *The output of the phase detector will be in saturation whenever the two inputs are not at exactly the same frequency*. By "saturation" we mean that the phase detector output will either stay close to the potential of the V_{cc} supply rail, or *ground*, depending on the polarity of the frequency error between the *A* and *B* inputs.

Finley's law can be stated in a simple way: *The phase detector "likes" to have the same frequency at its A and B inputs, and will take whatever action is necessary in order to maintain that condition.* The phase detector is the decision maker in the loop, and its output controls the VCO. The VCO output affects the frequency that the *A* phase detector input sees because of the feedback connection.

When a PLL frequency synthesizer is correctly operating (*locked*), the two phase detector inputs will *always* have the same frequency present!

Example 7-3

Calculate the frequency at points D, E, and C in the loop of Figure 7-11, given the following information:

f_{ref} = 1 kHz, and *N=2 (divisor)* and the loop state is locked.

Solution
Test point D is the reference frequency input, so by inspection, this frequency will be 1 kHz.

Test point E is calculated by using Finley's law. The phase detector will not be "satisfied" until both of its inputs are at the same frequency. The frequency present at the *bottom* phase detector input is already known; it is 1 kHz. Therefore, test point E must *also* be 1 kHz, because the phase detector will give the VCO voltage "commands" to make this so.

Test point C looks a little trickier. How can we find the VCO frequency? There is a divider circuit in between point C and point E. We know the frequency at point E is 1 kHz by Finley's law. The frequency at point C can be found by thinking "backwards" about the frequency divider. *If we have a divide-by-two divider and 1 kHz is coming out, what frequency must be coming in?* That's right -- the frequency at the divider input must be two times 1 kHz, or 2 kHz.

That's pretty cool! The VCO must be producing a frequency of 2 kHz in order for the divider to put out a frequency of 1 kHz (it is a divide-by-two circuit). The phase detector will not be satisfied until it sees 1 kHz at both of its inputs, and the only way that can happen is for the VCO to make 2 kHz. *The circuit has synthesized a 2 kHz signal from a 1 kHz signal!*

In Example 7-3 the VCO must be designed to be capable of producing a 2 kHz signal. This is a job for the engineer that designs the loop. If the VCO is incapable of producing the desired output frequency -- you guessed it, the loop will drop out of lock. This is highly undesirable!

A simple formula is often used to predict the output of a PLL synthesizer like the one in Figure 7-11:

(7-2) $f_{out} = N \Delta f_{ref}$

Where *N* is the divisor in the feedback loop, and f_{ref} is the applied reference frequency. This is not a particularly special formula; if you forget it, you can always find the output frequency of the PLL by using Finley's law, as we did in Example 7-3 above.

Example 7-4

Calculate the frequency at points E and C in the loop of Figure 7-11, given that f_{ref} = 10 kHz, and the following divisors:

a) N=89 b) N = 71 c) N = 100

Solution
a) Point E = 10 kHz, since f_{ref} = 10 kHz. (Finley's Law) Point C is the output node, and is computed using Equation 7-2: