Multi-Part Clock Synchronization Methods for Large Data Acquisition Systems

Doug La Porte

Introduction

High speed data conversion systems requiring a large number of low jitter clock signals often need to use multiple clocking devices. Compared to a clocking architecture requiring only a single clocking device, an architecture utilizing multiple devices complicates the system design in two major respects. First, these multiple devices must be synchronized to assure consistent clock phasing at all outputs. Second, maintaining low jitter is difficult as every device in the clocking system adds more noise. Designing a synchronized clocking system with both low jitter and a large number of outputs is a challenge.

Fortunately, Linear Technology's family of frequency synthesizers and clock distribution products addresses these concerns. For the purpose of this application note, it is convenient to sort all parts in the product family into one of three classes. The first part class has a full Phase Locked Loop (PLL) core and an integrated or external Voltage Controlled Oscillator (VCO) plus clock distribution with several frequency dividers and output drivers. For compact notation purposes, these PLL with clock distribution parts are referred to as PLL+Dist parts here. The second part class is similar with clock distribution and multiple frequency dividers and output drivers, but these parts do not have a PLL or VCO. These devices are referred to as Divider/Distribution, or Div/Dist, parts here. Last, some parts have neither a PLL nor frequency dividers but provide multiple output copies of the input signal and are referred to as fan-out buffers. Figure 1 summarizes these three part classes.

, LT, LTC, LTM, Linear Technology and the Linear logo are registered trademarks and EZSync, EZParallelSync, EZ204Sync, ParallelSync and LTC6951Wizard are trademarks of Analog Devices, Inc. All other trademarks are the property of their respective owners.

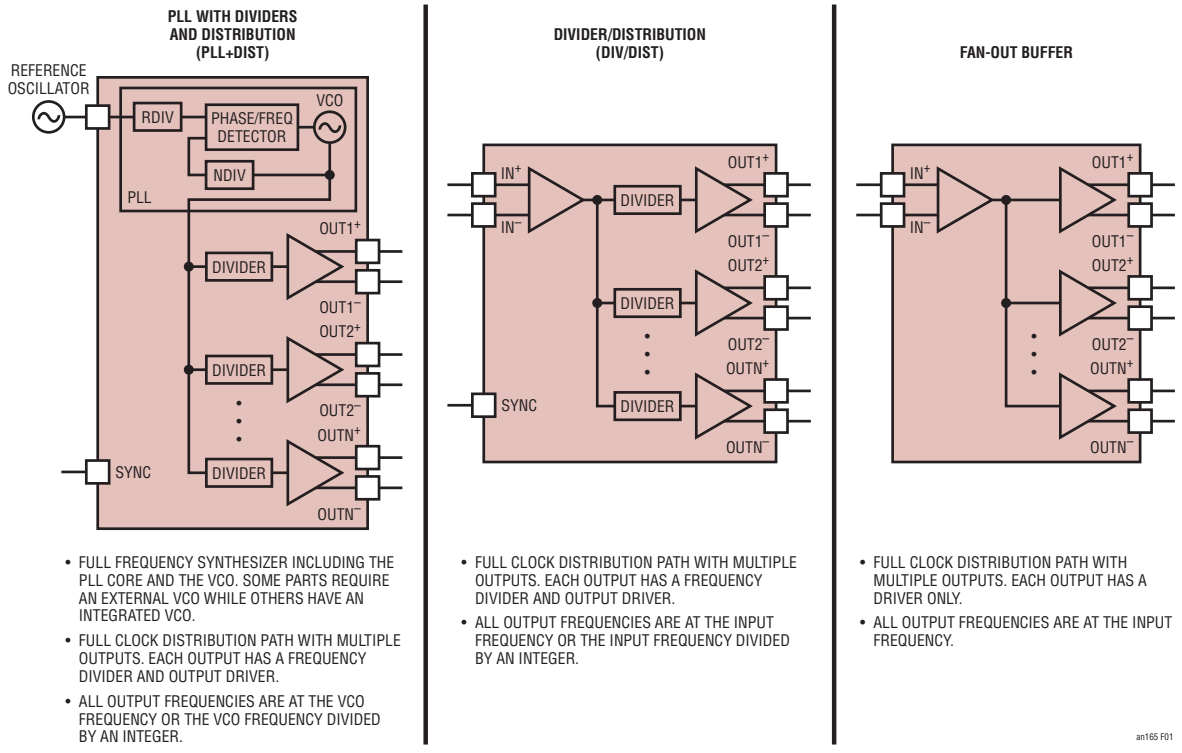


Figure 1. High Performance Clock Part Classes

Linear Technology addresses both the synchronization and jitter performance challenges by providing a full featured family of clocking products and three multi-chip synchronization methods. The three multi-chip synchronization methods are as follows:

EZSync™

EZSync is a simple way to generate synchronized clock outputs from multiple *cascaded* devices requiring only a simple logic signal or serial port interface (SPI) commands to achieve output signal alignment.

ParallelSync™

ParallelSync is a method to synchronize multiple PLL+Dist parts running in *parallel* driven by a common, reference clock *fan-out buffer network*. Synchronization is achieved through a common reference aligned signal.

EZParallelSync™

EZParallelSync is a simple way to synchronize multiple PLL+Dist parts running in *parallel* driven by a common reference clock *divider/distribution network*. Synchronization is easily achieved through simple logic signals or SPI commands.

A subset of EZParallelSync is EZ204Sync™ which is optimized to provide the device clocks and SYSREF signals required to achieve deterministic latency with JESD204B subclass 1 data converters in a lower noise and power efficient manner.

Figure 2 shows each of these synchronization topologies in block diagram form. Table 1 lists each topology along with comments on features and performance.

This application note provides an overview of each synchronization topology with performance trade-offs highlighted. Individual appendices follow providing a detailed description of the operation of each synchronization method. Further information on the explicit details of each specific clocking part is available in the part's data sheet and each part's clock synchronization guide.

Topologies: Clock Distribution versus Reference Distribution – General Considerations

EZSync is a clock distribution topology. ParallelSync and EZParallelSync are both reference distribution topologies. With EZSync, the beginning of the clock tree is a PLL+Dist part with multiple outputs. This first part is referred to as a controller as this part dictates and drives the synchronization protocol. The PLL+Dist portion of this part multiplies the low frequency reference oscillator (usually less than 200MHz) to a high frequency (usually greater than 1GHz). This high frequency signal is routed on-chip to the integrated frequency dividers and output drivers that then deliver either the VCO frequency or a divided down signal to multiple Div/Dist parts referred to as followers. The signals routed from the controller part to the follower parts are usually at a high frequency.

Synchronization of the controller part not only aligns all of the on-chip frequency dividers to each other, but also aligns these outputs to the input of the PLL's phase/frequency detector (PFD). Thus, when the loop is locked, the outputs have a consistent, repeatable relationship to the PFD and hence the reference clock.

Additionally, by staggering the synchronization signal's timing at each part, a follower part can act as a pseudo-controller thus providing infinite stages of expansion beyond the standard two stages. See Appendix A for the operational details.

The ParallelSync method starts with fanning-out the reference oscillator's signal to drive multiple PLL+Dist parts. The SYNC signal must be aligned with the reference clock and may need to be retimed into the reference clock domain and fanned-out to each of the several PLL+Dist parts. This retimed SYNC signal, SYNC-RT, must meet setup and hold timing requirements relative to the reference input signal at the PLL+Dist parts. The SYNC-RT signal synchronously resets all of the PLL+Dist parts' reference dividers and synchronizes each PLL+Dist part's outputs to its reference divider's output. In this way all of the outputs of all the PLL+Dist parts are all aligned. See Appendix B for the operational details.

EZParallelSync and EZ204Sync are similar to ParallelSync. The main difference is that the reference oscillator is not just fanned-out, it may also be divided down in frequency by a reference Div/Dist part or network of parts. Furthermore, each reference Div/Dist output may be divided down by a different number. Since all of these divided reference outputs must be synchronized, the reference Div/Dist path

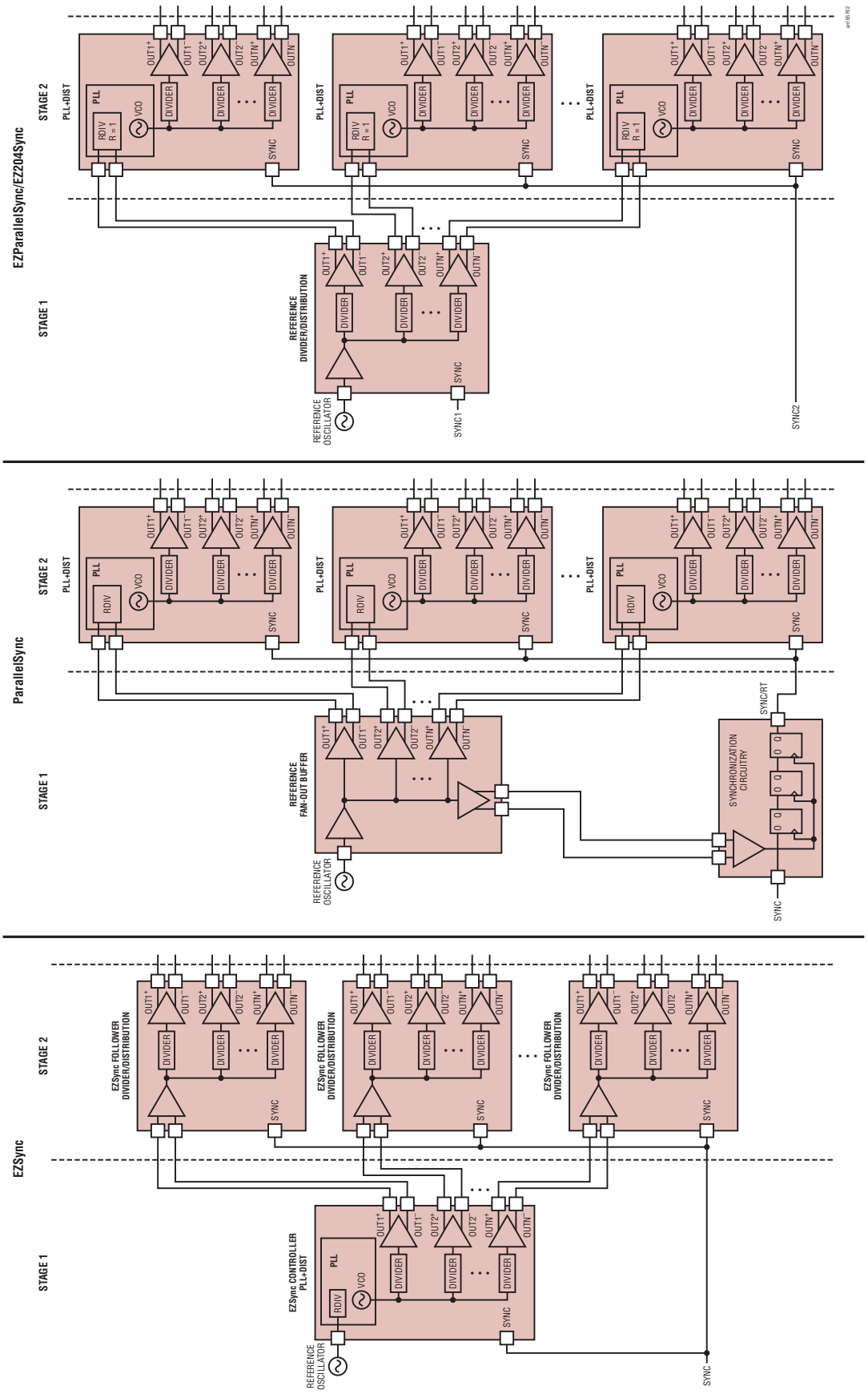


Figure 2. Sync Methods and Block Diagrams

is, when considered by itself, essentially an EZSync clock distribution type of topology.

These lower frequency divided reference signals are then routed to several PLL+Dist parts. Each of these PLL+Dist parts must have its reference divider set to one. In essence, the PLL+Dist part's reference divider has been figuratively relocated outside of the PLL+Dist part and into the reference divide/distribution area. Each of the PLL+Dist parts is synchronized as if it is an EZSync controller thus all outputs are aligned to the reference input clock. See Appendix C for the operational details.

Each of these three topologies has some advantages and some shortcomings. System cost, power consumption and Printed Circuit Board (PCB) layout considerations will evolve over time as newer clocking products come to market and must be considered on an individual basis for each application. However, there are several system level concerns and trade-offs that always apply when choosing between these topologies. The major trade-offs concern PCB signal routing issues, phase noise (jitter) accumulation through stages, frequency planning and synchronization issues.

Topologies: Clock Distribution versus Reference Distribution – Signal Routing Considerations

One key distinction between clock distribution and reference distribution is the nature of the signals that are expanded and routed long distances on the PCB. The clock distribution topology routes high frequency signals derived from the PLL+Dist controller part. The reference distribution topology routes lower frequency signals at the reference oscillator frequency or lower.

Routing high frequency signals requires more care than routing lower frequency signals. Each signal is typically differential and must be routed as a differential transmission line pair with the line properly terminated to avoid ringing and excessive signal distortion. Lower frequency signals are a bit more forgiving of transmission line impedance and termination mismatch.

Additionally, routing high frequency signals long distances may necessitate the use of an expensive RF type of PCB material to keep signal loss to an acceptable level. By using a reference distribution topology, the lower frequency signals run the long distance and the high frequency signal paths are kept short so standard PCB materials may be used.

High frequency signals also radiate more efficiently than lower frequency signals thus coupling more interfering energy onto neighboring circuitry. However, by setting the clock signal's frequency very high, radiated interference from the clock and its harmonics may be placed well above the signal band of interest and are thus easily filtered out. This is not always practical or possible however.

One last signal routing difference between the topologies is whether or not the signals can be AC-coupled. For the EZSync method, all interstage connections (e.g. between Stage 1 and Stage 2) must be DC-coupled. For the ParallelSync method, all interstage clock signals may be AC-coupled, however, the SYNC-RT signal must be DC-coupled. For the EZParallelSync method, all interstage connections may be AC-coupled.

Topologies: Clock Distribution versus Reference Distribution – Phase Noise (Jitter) Accumulation

Another key distinction between clock distribution and reference distribution is how phase noise, or jitter, accumulates when adding more stages of expansion. All active circuitry, and many passive circuits, add noise. Every device in the clocking signal path adds noise.

With the clock distribution topology, each stage adds broadband noise that has a minor effect on the close in phase noise (low offset frequency), but a significant effect on the wideband phase noise floor. With the reference distribution architecture, only the close in phase noise is affected with minimal effect on the total jitter at the final stage of expansion.

In most cases, the reference distribution topology gives the best phase noise performance and hence the lowest jitter. To see why, consider how a PLL reacts to noise on the reference clock signal. A PLL is simply a negative feedback servo system that produces an output frequency at a multiple of the reference frequency. A PLL provides frequency "gain" much like an op amp provides voltage gain. Like all servo systems, including op amps, a PLL has a limited closed loop bandwidth. Within the bandwidth of the PLL, the output responds directly to all input signals, including

Table 1

	EZSync	ParallelSync	EZParallelSync / EZ204Sync
Topology	**Clock Distribution Topology** One main PLL+Dist part with cascaded divider/ distribution parts.	**Reference Distribution Topology** Cascaded reference fan-out distribution parts driving multiple parallel connected PLL+Dist parts.	**Reference Distribution Topology** Cascaded reference divider/distribution parts driving multiple parallel connected PLL+Dist parts.
Sync Signaling	Sync requires a single simple logic signal or serial port commands.	Sync requires an reference clock aligned signal. There is no serial port option.	Sync requires separate logic signals or serial port commands for each stage of the design.
Sync Timing Requirement	Easy	Moderate	Easy
Stage 2 Parts May Be Powered Down, Powered Back Up and then Easily Resynchronized.	No	No	Yes
Outputs Are First Edge Aligned	Yes	Yes	No
Outputs Are Steady State Phase Aligned	Yes	Yes	Yes
Signal Routing and Coupling	High frequency clock signals routed long distances on PCB. Stage 1 to Stage 2 signals MUST be DC-coupled.	Low frequency reference signals routed long distances on PCB. High frequency signal routing is kept as short as possible. Stage 1 to Stage 2 signals may be DC- or AC-coupled, however, the SYNC-RT signal MUST be DC-coupled.	Low frequency reference signals routed long distances on PCB. High frequency signal routing is kept as short as possible. Stage 1 to Stage 2 signals may be DC- or AC-coupled.
Phase Noise/Jitter Accumulation	Each cascaded stage adds wideband phase noise.	Each cascaded reference stage adds only close-in phase noise. Wideband phase noise dominated by the PLL+Dist parts.	Each cascaded reference stage adds only close-in phase noise. Wideband phase noise dominated by the PLL+Dist part.
Jitter Performance	Good	Best	Best
Frequency Planning notes	The PLL+Dist controller part's VCO is an integer multiple of the PLL's reference divider (RDIV) output frequency. All outputs are at the VCO frequency or the VCO frequency divided by an integer.	All of the PLL+Dist parts must set their reference divider (RDIV) values such that all of the RDIV output frequencies are the same. In other words, all PLL+Dist parts must have the same phase/ frequency detector (PFD).	All output frequencies from the reference divider/ distribution network must be an integer multiple of the lowest output frequency from the network. All PLL+Dist parts must set their reference divider to one (R = 1). Further, all outputs from each PLL+Dist part must be integer multiples of that PLL+Dist part's Reference input frequency.

noise. At frequencies beyond the PLL's bandwidth, input signals (and noise) are progressively attenuated as the servo cannot track these signals. At very high frequency, the output noise is dominated by the VCO's noise and the output driver's noise floor. With a limited input response bandwidth, a PLL filters broadband noise acting much like a very narrow bandpass filter.

As an illustrative example, consider the circuits shown in Figures 3 and 4. These figures show how to use the LTC6954 Div/Dist part and the LTC6951 PLL+Dist part

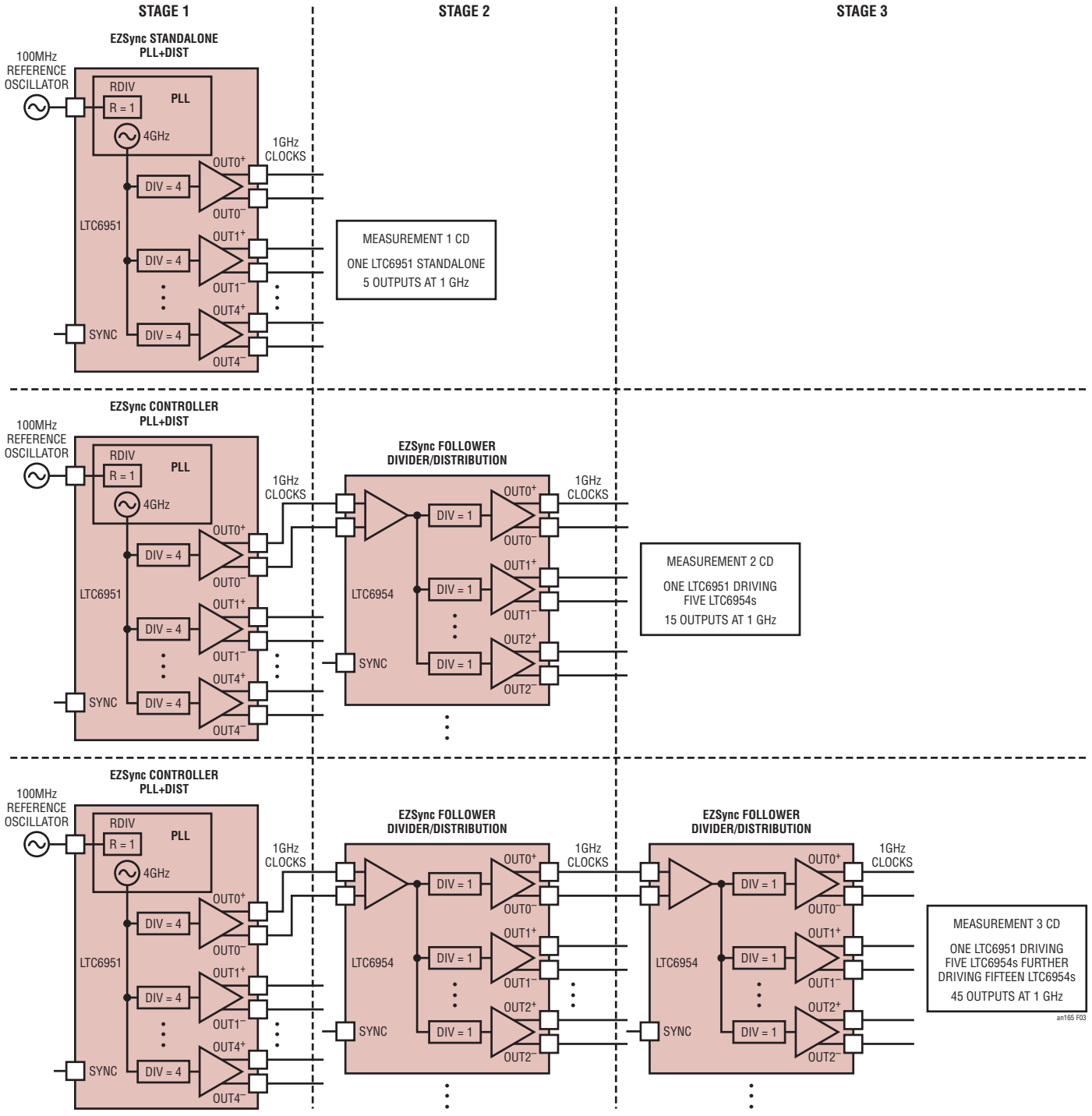


Figure 3. Clock Distribution through Multiple Stages

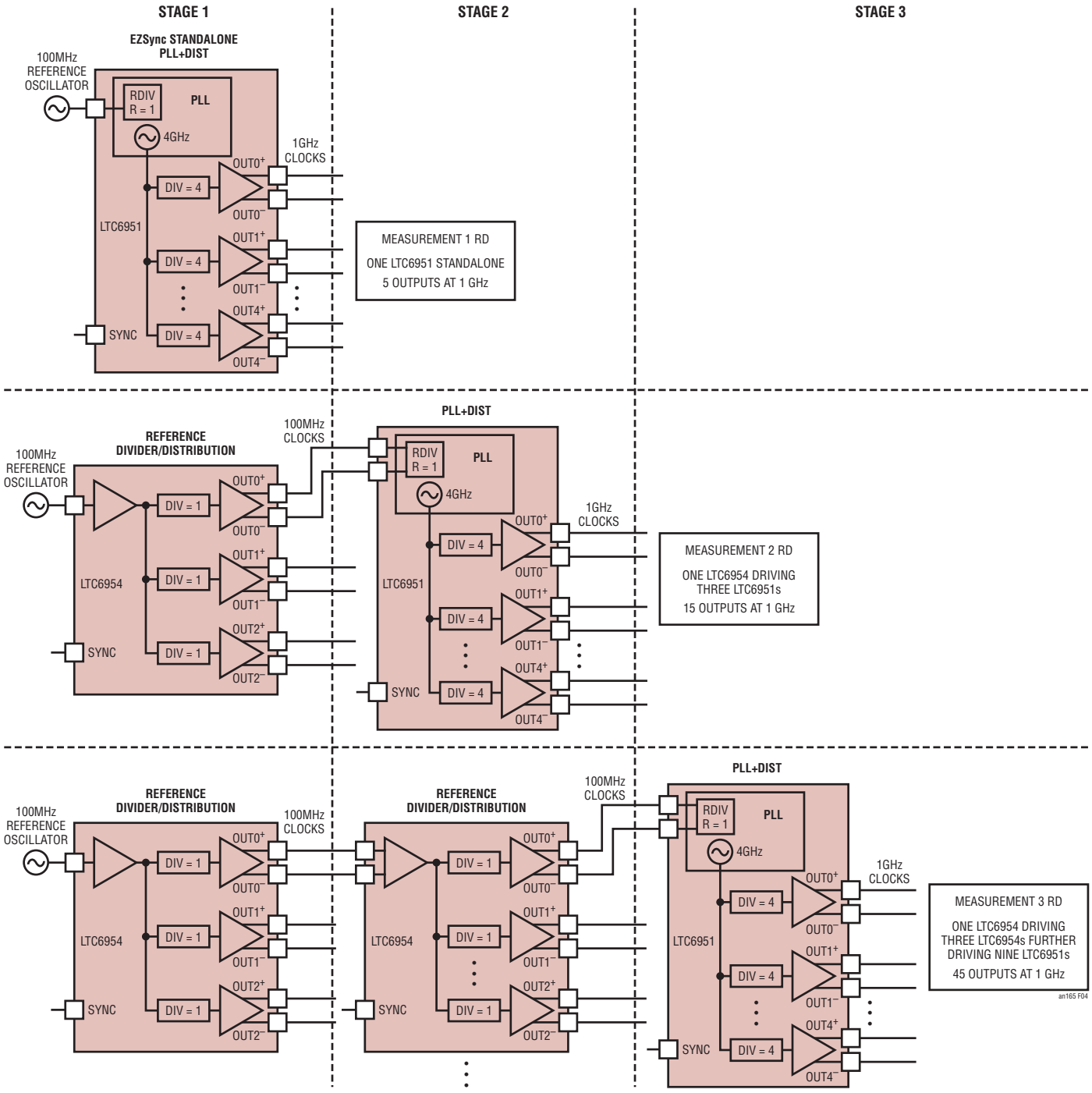


Figure 4. Reference Distribution through Multiple Stages

to multiply the reference oscillator's 100MHz signal and deliver multiple outputs at 1GHz. A single stage gives 5 outputs. Using two stages gives 15 outputs while three stages deliver 45 output signals. Note that this is a simple, illustrative example using two of Linear Technology's smaller clocking parts. Using some of our larger parts a three stage design can support over one thousand outputs.

Figure 3 uses a clock distribution topology while Figure 4 uses a reference distribution topology. For simplicity, all of the dividers of the LTC6954 parts are set to one although

they could be any number. All of the LTC6951 parts are set to deliver a 1GHz signal on each output although these could be different frequencies.

For each topology the output phase noise is measured at each stage of expansion. Figure 5 shows the measured phase noise as well as the calculated jitter for all three stages of each topology.

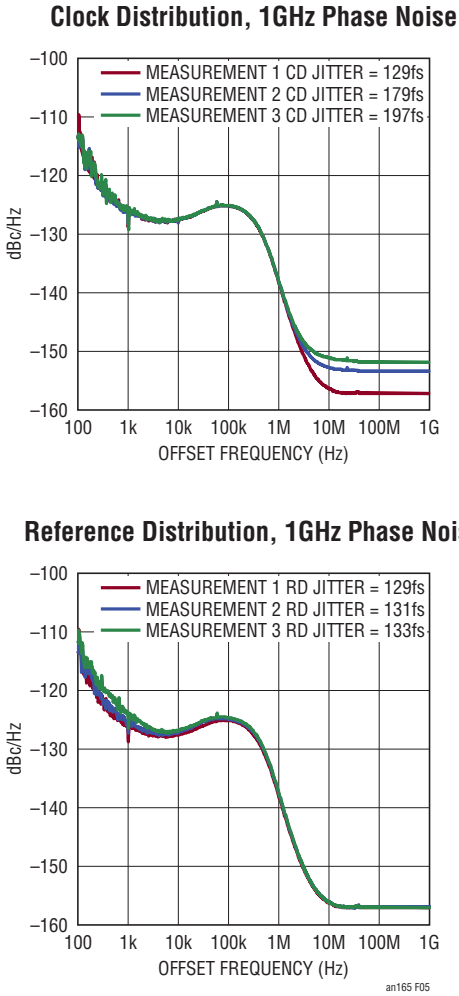


Figure 5. Measured Phase Noise and Jitter for the Clock Distribution and Reference Distribution Circuits Shown in Figures 2 and 3

The plot at the top of Figure 5 shows the measured results from the clock distribution topology circuits of Figure 3. This plot clearly illustrates how the close in phase noise is minimally affected, but the broadband phase noise floor rises with each stage added. Converting the phase noise frequency domain data into a time domain jitter number requires integrating the phase noise. The signal is at 1GHz, so the integration limits are from 100Hz to 1GHz. The 100Hz lower integration limit is somewhat arbitrary but the upper limit must be 1GHz for a 1GHz signal to calculate the true total jitter as a data converter sees it. While the close in phase noise is 20dB to 30dB higher than the broadband phase noise floor, it only extends in frequency to about 1MHz. The phase noise floor accounts for the remaining 999MHz of integration bandwidth. Raising the phase noise floor by 3dB to 6dB adds substantial noise and is the dominant factor in the 68fs jitter increase from 129fs at stage 1 to 179fs at stage 2 to finally 197fs at stage 3.

The phase noise floor of the LTC6954 is a few dB higher than that of the LTC6951. If the LTC6954 had the lower phase noise floor than the LTC6951, the increase in the total phase noise floor and the resulting jitter would not have been as great. This highlights the importance of using Div/Dist follower devices with a low phase noise floor to achieve low jitter clock signals at the final stage.

For the clock distribution topology it is also possible to easily calculate the final stage's total output jitter directly with jitter numbers. The *absolute* jitter of the PLL+Dist part in the first stage is summed with the *additive* jitter (sometimes referred to as residual jitter) of the following Div/Dist stages in a root sum squared manner as shown in Equation 1.

$$JITTER_{TOTAL} = \sqrt{\begin{array}{l}(ABSOLUTEJITTER_{PLL+DIST})^2 \\ +(ADDEDJITTER_{DIV+DIST})^2 \\ +(ADDEDJITTER_{DIV+DIST})^2\end{array}}$$

To estimate the absolute jitter for the PLL+Dist parts, use the appropriate Linear Technology Wizard program (e.g. LTC6951Wizard™ for the LTC6951 part). Linear Technology's Wizard programs are CAD tools that simulate the expected phase noise for a given application and are freely available to download at the Linear Technology website. To determine the added phase noise of any of the Div/Dist parts, consult the data sheet for each part.

The lower plot in Figure 5 shows the measured results from the reference distribution topology circuits of Figure 4. The PLL's filtering effect is evident here as only the very close in phase noise increases with additional stages of expansion. With this noise increase being small and over a relatively small bandwidth, the jitter increases by only 4fs

from 129fs at stage 1 up to 133fs at stage 3. The increase is so small that it is hardly worth noting.

Note that these examples are for illustration only. The performance of any given application will be different in absolute numbers, but the basic concepts always apply.

Also note that a clocking solution with more expansion stages does not necessarily have more jitter than a solution with a single part. The clock distribution example as shown in Figure 3 delivers 45 outputs at 1GHz with 197fs of true broadband jitter. This appears to be poor when compared to the 133fs performance of the reference distribution example. However, there are numerous products on the market that do not come close to delivering even 200fs from a single part with far fewer than 45 outputs. Cascading several low noise parts still delivers lower phase noise and jitter than a single mediocre part.

Topologies: Clock Distribution versus Reference Distribution – Frequency Planning

Frequency planning must be considered when choosing between a clock distribution and reference distribution topology. In general, neither approach has a large advantage in flexibility here, but the way to achieve the system's desired output frequency requirements is different.

The clock distribution topology is simple and straightforward. With clock distribution, the PLL+Dist controller part's VCO is at the highest frequency in the system. All output signals are either at the VCO frequency or this frequency divided by an integer number.

With the reference distribution topology using the ParallelSync method, developing a frequency plan is also quite straightforward with only one simple rule. With the ParallelSync method, all of the PLL+Dist part's phase frequency detector (PFD) frequencies must be the same. As all of the PLL+Dist part's reference input frequencies are also the same this results in all of the PLL+Dist part's having the the same reference divider (RDIV) setting.

With the reference distribution topology using the EZParallelSync method, developing a frequency plan is a bit trickier. There are two simple rules that must be followed.

First, there are constraints on the PLL+Dist part's phase/frequency detector (PFD) frequency. When using the EZParallelSync method, all of the PLL+Dist parts' PFD frequencies must be harmonically related. Stated another way, all of the PLL+Dist parts' PFDs must operate at the minimum PFD frequency of the system or an integer multiple of this minimum PFD frequency. This requirement may be expressed by the following equation:

$$f_{PFD} = f_{PFD(MIN)} \bullet K$$

Where f_{PFD} is the frequency of the specific PLL+Dist part's PFD being considered, $f_{PFD(MIN)}$ is the lowest PFD frequency from all PLL+Dist parts in the system and K is an integer greater than or equal to one. Note that with the EZParallelSync method each PLL+Dist part's RDIV is set to one (R = 1) so the PFD frequency is the same as the RDIV input frequency.

The second rule is that each PLL+Dist part's output frequencies must be integer multiples of that PLL+Dist part's PFD frequency. This requirement may be expressed by the following equation:

$$f_{OUT} = f_{PFD} \bullet L$$

Where f_{OUT} is the frequency of the specific PLL+Dist part's output being considered, f_{PFD} is the frequency of the PFD for the specific PLL+Dist part associated with the output being considered and L is an integer greater than or equal to one.

These EZParallelSync reference distribution rules may seem to limit the usefulness of this topology. However, in practice the output frequencies that are achievable using reference distribution are actually greater than with a clock distribution system.

Synchronization Considerations with EZSync, ParallelSync and EZParallelSync

Synchronization of all of the part's frequency dividers is very important. Since every frequency divider is initially programmed one part and one register at a time through the serial port interface (SPI) and not in any way synchronized with the VCO or clock signals, the outputs will be in random phase relationship every time the system is powered up or programmed.

For example, consider two dividers each set to divide by four and programmed through the SPI. As shown in figure 6 there are four possible phase relationships between the two output's dividers: in phase, lagging by ¼ cycle, anti-

phase and leading by ¼ cycle. Without synchronization, the system will have to tolerate this unpredictability.

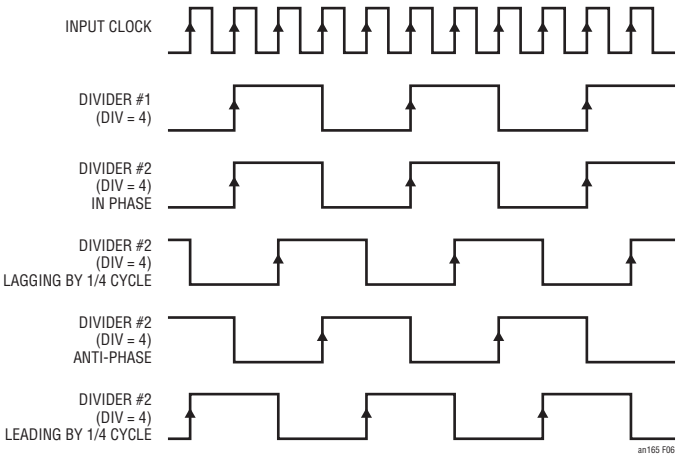


Figure 6. Divider Outputs Possible Phasing Prior to Synchronization

There are many systems where the correct alignment of all output clocks is vital for proper operation. However, even in systems where the need does not appear to be vital, it is still good practice to synchronize all outputs. Without synchronization, problems may occur only occasionally and can be difficult to track down. Assuring a consistent alignment of all clock signals avoids odd, occasional race conditions or interference states.

It is important to note what synchronization does and does not provide. The primary purpose of synchronization is to assure that the output signals are all consistently phase aligned to each other the same way every time the system is initialized. This is often referred to as having a deterministic condition. Synchronization does not assure absolute time alignment of clock edges. While the outputs' timing skew of each part is very small, PCB trace length matching or cable length matching and other routing concerns still apply. Additionally, propagation delay through each part is controlled as well as possible, but part-to-part variation and changes over temperature must be considered.

All three of Linear Technology's multi-part synchronization methods provide the ability to synchronize all outputs and provide a reliable deterministic condition. There are two major synchronization considerations though.

First, some synchronization methods require a logic signal to initiate synchronization while others allow for synchronization to be done through SPI commands alone. EZSync can use either a single logic signal, multiple logic signals or SPI commands. Similarly, EZParallelSync (or EZ204Sync) requires several logic signals or SPI commands to initiate and complete clock synchronization. However, ParallelSync requires a reference clock aligned logic signal that must meet the PLL+Dist part's input set-up and hold requirements. SPI controlled synchronization is not an option for ParallelSync.

Second, some synchronization methods deliver outputs that are first edge aligned, while other methods are not aligned on the first edge, but are steady-state phase aligned. As shown in Figure 7, first edge aligned means that all outputs are held low during the synchronization initialization and, once synchronization is completed, make a clean transition from low to high on the same input clock rising edge. Thus the first rising edge of all clock outputs are coincident and correctly aligned regardless of the divide setting. EZSync and ParallelSync provide first edge synchronization. EZParallelSync does not support first edge synchronization.

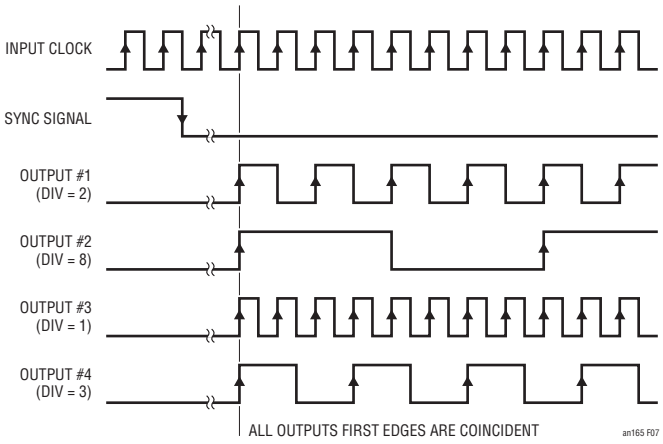


Figure 7. First Edge Synchronization Assures that the First Rising Edge of All Outputs Are Coincident Regardless of the Divide Number

As shown in Figure 8, steady-state phase alignment means that the initial first edges are not coincident across all parts, but all outputs are consistently in the same phase alignment once the synchronization process is fully completed.

Few applications require first edge alignment. In most cases assuring that all signals consistently have the same steady state phase relationship after completion of the synchronization process is sufficient.

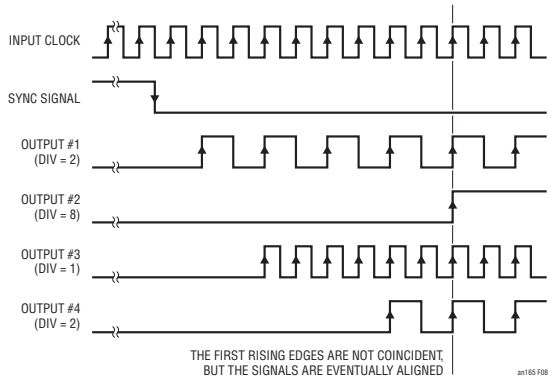


Figure 8. Steady-Stage Synchronization Assures Consistant Output Signal Phasing Regardless of the Divide Number Although the First Rising Edges Are Not Aligned

Hybrid topologies

Combinations of these synchronization topologies are also possible. A hybrid topology may make sense for a given application based on the number of output frequencies and signals required or due to natural circuit partitioning between multiple PCBs.

As noted earlier, EZParallelSync has a reference distribution section that, considered by itself, is essentially an EZSync clock distribution topology. Additionally, the following PLL+Dist part can also be an EZSync controller part. As shown in Figure 9, this Stage 3 PLL+Dist EZSync controller may be followed by additional EZSync follower devices to provide even more output signals. This resulting circuit is part reference distribution, part clock distribution. This circuit may be further expanded by either adding more stages of reference distribution between Stages 2 and 3 or by adding more stages of EZSync followers after Stage 4.

Summary

Data conversion systems requiring a large clocking system with both low jitter clock signals and a large number of output signals presents a daunting challenge. The process of determining the best circuit topology and which parts to use in a given system is application dependent and requires making trade-offs. Beyond the determination of how many signals and which frequencies are required there are issues of how to partition the system design. In some cases there may be a main board connected to several secondary boards where minimizing the number of signals in the back plane or cable harness is a high priority. Other systems may place power consumption, PCB area or cost as a high priority. Each system will be unique.

The synchronization options covered here, combined with Linear Technology's family of low jitter frequency synthesizers and clock distribution products, allows an optimal solution for every system need.

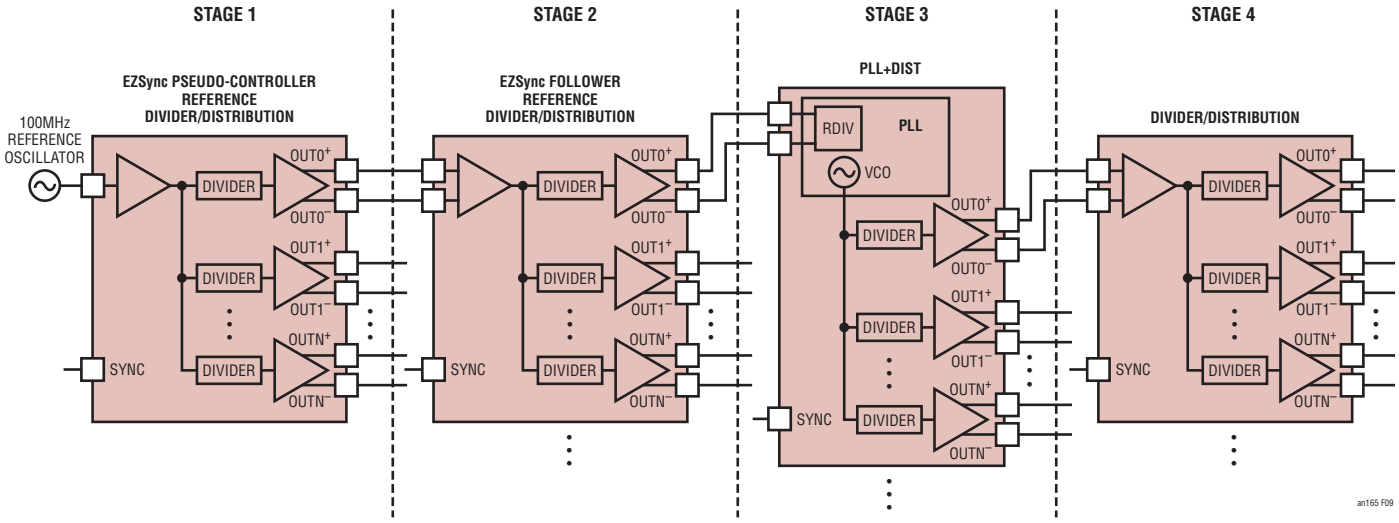


Figure 9. Hybrid Topology Created by Adding an EZSync Follower Stage After the PLL+Dist EZSync Controller in Stage 3

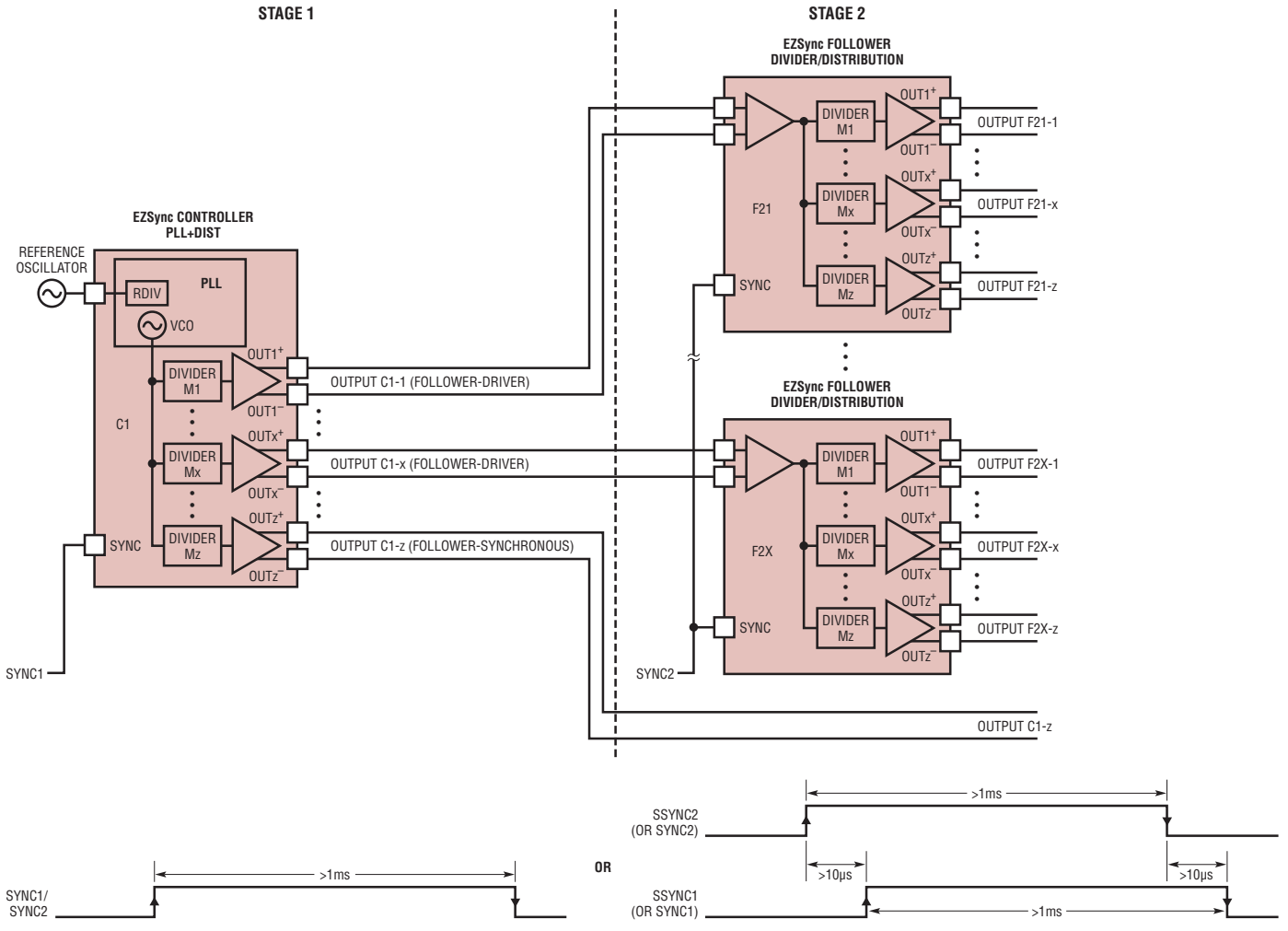
APPENDIX A: EZSync DETAILS

Overview and Terminology

EZSync is a *clock distribution* topology and is a simple way to generate synchronized clock outputs from multiple *cascaded* devices requiring only a simple logic signal or serial port interface (SPI) commands to achieve synchronization.

Many multi-part clock systems have one part at the front of the clock tree that is in control of synchronization acting much like the conductor of an orchestra. For the EZSync protocol we call this "conductor" part the CONTROLLER. The CONTROLLER part runs all of the timing and signaling of synchronization events. The CONTROLLER achieves synchronization management through precisely gating on and off its output clock signals in a consistent manner as defined in the EZSync protocol. Runt pulses are completely avoided. To achieve synchronization, the end user is required to perform a few simple, loosely timed tasks as the CONTROLLER part does all of the hard work.

As shown in Figure A1, the CONTROLLER part connects to one or more EZSync FOLLOWER parts. Each FOLLOWER part receives a gated clock signal from the CONTROLLER and responds appropriately to achieve consistent clock signal synchronization every time. As the input clock signal



SYNC1 AND SYNC2 MAY BE TIED TOGETHER WITH PCB ROUTING ASSURING <10µs TIMING SKEW AT THE PARTS' SYNC INPUTS

WHEN USING SPI CONTROLLED SOFTWARE SYNCHRONIZATION CONTROL (SSYNC), SIMULTANEAOUS SIGNALING IS NOT POSSIBLE. IN THIS CASE, THE SSYNC SIGNALS MUST BE STAGGERED IN TIMING WITH THE STAGE 2 DEVICES SIGNALING PRECEEDING THE STAGE 1 SIGNALING. SSYNC SIGNALING WITHIN STAGE 2 CAN HAPPEN IN ANY ORDER AS LONG AS THE LAST PROGRAMMED PART IN STAGE 2 IS PROGRAMMED BEFORE THE STAGE 1 PART IS PROGRAMMED

an165 A01

Figure A1. Sample Block Diagram for a Two Stage EZSync Design. This Is Just One Simple, Illustrative Example. There Are Hundreds of Possible Configurations

to each FOLLOWER part is gated off and on, the connection between the CONTROLLER and the FOLLOWER parts must be DC-coupled.

All of Linear Technology's PLL with clock distribution parts (PLL+Dist) are CONTROLLER parts with *some* of these parts also capable of operating as a FOLLOWER. Additionally, all PLL+Dist also operate in a STANDALONE mode when used alone and not driving FOLLOWER parts. *All* of Linear Technology's Divider/Distribution parts (Div/Dist) are FOLLOWER parts. As will be described later, by staggering the timing of the synchronization signals, a FOLLOWER part can also perform the CONTROLLER function. When a FOLLOWER is used in this manner we refer to it as a PSEUDO-CONTROLLER. Consult each part's data sheet for details of the part's capabilities. Note that fan-out buffers do not alter the frequency of their input signals (no frequency multiplication or division), so there is no need for synchronization with this class of parts.

In normal operation synchronization is idle. EZSync synchronization is a two phase process: initialization and completion. The initialization phase must be finished before starting the completion phase. Once the completion phase is finished, synchronization is again idle.

On each part, the synchronization state (idle, initialization or completion) is controlled by either a dedicated pin driven by external logic or a serial port register bit controlled through a SPI write command.

All Linear Technology parts have a pin allocated for synchronization control. This pin is typically controlled by a simple CMOS logic signal although some parts support differential signaling. The pin is typically labeled SYNC or possibly EZS-SRQ in the case of some parts where this pin can perform multiple functions. For simplicity this pin will be referred to as SYNC here.

In addition to the SYNC pin, *most* Linear Technology parts also have a serial port register bit for synchronization control. The exceptions are the LTC6950 and LTC6954 parts. This register bit is simply labeled SSYNC (shorthand for Software SYNC) or possibly SSRQ in the case of some parts where this bit can perform multiple functions. For simplicity this bit will be referred to as SSYNC here.

In normal operation (when synchronization is idle) the SYNC pin and the SSYNC bit are both at a logic low ("0"). To start the initialization phase of a synchronization event, the user must either drive the SYNC pin to a logic high or write "1" to the SSYNC bit through the SPI. Note that internal to each part the SYNC and SSYNC signals are logically combined through an OR gate to form one signal. Each application should use either the SYNC or SSYNC signal with the other signal held to a logic low ("0") state throughout that entire synchronization process. Internal to the part, the ORed SYNC/SSYNC signal is retimed to the input signal (the VCO signal on PLL+Dist parts) through multiple flip-flops to avoid any metastability issues. On PLL+Dist parts this signal is additionally retimed to the output of the PLL's feedback divider (NDIV). This SYNC/SSYNC rising edge, low to high ("0" to "1"), transition starts the initialization phase of synchronization.

During the initialization phase all of the parts' output dividers are placed into a state where they continue to operate normally until each divider's output naturally transitions from a logic high to low. Once the output goes low, it stays frozen at the low state and ignores the input clock signal. The CONTROLLER part will deliver clock signals to the FOLLOWER parts for a long enough period of time to ensure that all FOLLOWER parts' dividers have reached the logic low output state. A short time later, the CONTROLLER's outputs will go through the same process until all output divider outputs from all parts are frozen in the logic low state. Once this state is achieved the initialization phase is finished. SYNC (SSYNC) must be held high for the full duration of the initialization phase which must be at least 1ms. It may be much longer if desired but cannot be less than 1ms.

To start the completion phase of an EZSync synchronization event, the user must drive the SYNC (SSYNC) signal to a logic low ("0") state using the same signal used in the initialization phase (SYNC or SSYNC with the unused signal still held low). Internal to the part, the signal is again retimed to the input signal (the VCO signal on PLL+Dist parts) through multiple flip-flops to avoid any metastability issues. On PLL+Dist parts the signal is again additionally retimed to output of the PLL feedback divider (N divider). This SYNC/SSYNC falling edge, high to low ("1" to "0"), transition starts the completion phase of synchronization.

During the completion phase all of the FOLLOWER parts are first armed and ready to receive clock signals. The CONTROLLER part then synchronizes all of its on-chip dividers and starts delivering clock signals. All FOLLOWER

parts will ignore the first seven input clock rising edges and begin operating normally on the eighth input clock rising edge. Functionally, any device operating as a FOLLOWER will propagate its input clock signal with a seven cycle delay.

While the behavior of the FOLLOWER parts is fairly simple, the timing of the CONTROLLER's output signals is quite complicated. Fortunately the EZSync CONTROLLER part takes care of most of the issues.

Each of the CONTROLLER's individual output divider/drivers can be independently configured into one of three general operating modes:

1. Synchronization Disabled
2. Follower-Driver
3. Follower-Synchronous

Some parts have other additional modes for special signaling such as SYSREF signal generation for JESD204B clocking systems. These modes are useful in many systems, but do not affect multichip clock synchronization.

A CONTROLLER part's output configured with synchronization disabled is the simplest to describe. In this mode, this individual output is not affected at all by the SYNC or SSYNC signals. This allows the output to run continuously, uninterrupted by any synchronization event. This feature is useful for non-critical clocks that don't need to be aligned with the data acquisition clock signals and may not tolerate being gated off for a short period of time. An example might be general purpose FPGA or microprocessor clocks. Individual outputs on both CONTROLLER and FOLLOWER parts each have a serial port register bit that controls whether or not synchronization is enabled or disabled. This bit is usually labeled SYNC_ENx or SYNCENx where the "x" is the number of the part's individual output.

A CONTROLLER part's output configured for Follower-Driver operation is expected to provide the input clock to a FOLLOWER part. Each of the many CONTROLLER part's outputs may have different divider settings, thus potentially delivering different output frequencies to each FOLLOWER device.

A CONTROLLER part's output configured for Follower-Synchronous operation is expected to provide a clock signal that is aligned to the *output* of the FOLLOWER parts.

To deliver multiple CONTROLLER and FOLLOWER outputs at multiple output frequencies from the CONTROLLER and multiple FOLLOWER devices that achieve consistent, synchronization is fairly complicated. Thankfully, the EZSync method greatly simplifies this task by utilizing each part's individual output's programmable *clock cycle* delay feature. By selecting the correct clock cycle delay and pulsing the SYNC pin or the SSYNC bit, reliable clock synchronization is easily achieved.

Setting the Individual Output Cycle Delays to Achieve Synchronization of a Two Stage Design

Setting the correct clock cycle delays for each output can seem a bit daunting at first. However it is really not that difficult if the procedure is broken up into a few simple steps. This section covers delay setting for a two stage design with higher stage design considerations covered in the next section.

Some EZSync CONTROLLER parts (like the LTC6950) have a built-in feature that automatically sets the correct cycle delay value for each output when the user labels the output as operating in the Follower-Driver or Follower-Synchronous mode. This output labeling is done by setting the serial port register bit FLDRVx (where "x" is the specific output number) to "1" for Follower-Driver mode or "0" for Follower-Synchronous mode. For these parts, this is all that the user needs to do.

Other EZSync CONTROLLER parts do not have a dedicated FLDRVx bit and rely on the user to program the input clock cycle delays to the correct settings to assure that all outputs are synchronized. To determine the correct cycle delay settings, the first step is to determine the maximum output divider setting for *all* CONTROLLER Follower-Driver outputs (those outputs providing the input clock to a FOLLOWER part). Call this value $M_{CFD(MAX)}$. The appropriate delay for each Follower-Driver output is calculated by equation A1.

$$DELAY_{CFDx} = (M_{CFD(MAX)} - M_{CFDx}) \bullet 7 \qquad (A1)$$

Where "M_{CFDx}" is the divider value and "$DELAY_{CFDx}$" is the cycle delay setting for a specific Follower-Driver output. Simply calculate the cycle delay for each individual Follower-Driver output and program this value into the appropriate serial port registers.

A CONTROLLER output configured for Follower-Synchronous operation is expected to provide the clock input

directly to a data converter or some other device and achieve consistent, synchronization with the *outputs* of FOLLOWER devices. On parts with the built-in Follower-Driver/Synchronous mode setting feature all that the user needs to do is to write a "0" to the FLDRVx register bit (where "x" is the specific output number). For other parts, the correct delay setting is calculated by equation A2.

$$DELAY_{CFSx} = M_{CFD(MAX)} \bullet 7 \quad (A2)$$

Where "$DELAY_{CFSx}$" is the cycle delay setting for a specific Follower-Synchronous output. Simply calculate the cycle delay for each individual Follower-Synchronous output and program this value into the appropriate serial port registers.

Note that all of these cycle delay settings will result in all outputs aligned to the same clock cycle. If the application calls for outputs to be offset by one or more cycles, simply add or subtract the one or more cycles to the delay determined by equation 1 or 2. Also note that all of the above settings guarantee that all outputs from all devices will be synchronized to the same CONTROLLER input clock edge (the VCO edge for PLL+Dist parts). Device propagation delay and PCB routing delay is not compensated for. EZSync achieves consistent and repeatable clock *cycle* alignment. To achieve absolute *time* alignment of the outputs, many parts have additional *time delay* circuitry that can be used to fine tune output signal timing. This time delay is often called analog delay or ADLY which is different from *cycle delay*.

A sample block diagram for a two stage design is shown in Figure A1 and a sample timing diagram showing the SYNC (or SSYNC) signal timing and output waveforms is shown in Figure A2.

Expansion Beyond Two Stages

A design with two stages can support 15 outputs with the smaller parts in the Linear Technology clock distribution family and as many as 121 outputs with our larger parts. This is usually more than sufficient for any application. There are some systems that may require more than 121 output *signals*, but it is extremely unlikely that there is a need for 121 unique output *frequencies*.

With this in mind, most very large systems use simple fan-out buffers that do not require synchronization. Fan-out buffers may be placed in between the CONTROLLER and FOLLOWER parts to provide multiple copies of the follower-driver or follower-synchronous signals. In this case the fan-out buffer's input and output signals must be DC-coupled for the EZSync protocol to function properly. Fan-out buffers may also be used after the FOLLOWER parts to provide multiple copies of FOLLOWER output signals and may be AC-coupled if first edge alignment is not required.

Should a system need a very large number of output frequencies or should it make sense to partition the clocking architecture so as to require more than two stages, EZSync can still deliver a good solution. The procedure to set all of the correct cycle delays for correct synchronization can get quite complicated and is beyond the scope of this document. Contact Linear Technology's applications department for assistance. A sample block diagram for a three stage design is shown in Figure A3 and a sample timing diagram showing the SYNC (or SSYNC) signal timing and output waveforms is shown in Figure A4.

SYNC or SSYNC Signal Transition Timing

Each CONTROLLER part carefully manages several timing events in order to achieve clock synchronization. However, the user only needs to program the register values described above and provide the SYNC or SSYNC signals. There are a few timing requirements regarding the SYNC and SSYNC signal. These requirements are not difficult to achieve, but they are important.

For a simple two stage design utilizing the SYNC pins, all of the SYNC pins may be tied together (the FOLLOWERS and the CONTROLLER) and driven by one logic signal. The SYNC signal should start at a logic low state. To perform a synchronization, simply pulse the SYNC signal to a logic high state for a minimum of 1ms and return the signal to a logic low state. The 1ms timing is not critical. It may be much longer, but not shorter than 1ms. Once the EZSync synchronization event is complete, the SYNC pins should continue to be held low during normal operation.

The only additional requirement here is that there cannot be too much skew in the arrival of the SYNC signal at the CONTROLLER and FOLLOWER parts. The skew between when the signal arrives at the CONTROLLER and when the signal arrives at the FOLLOWER must be less than ±10µs. This should be easy to achieve with standard logic and reasonable trace lengths. (Note: 10µs is the approximate delay for 2km of cable!)

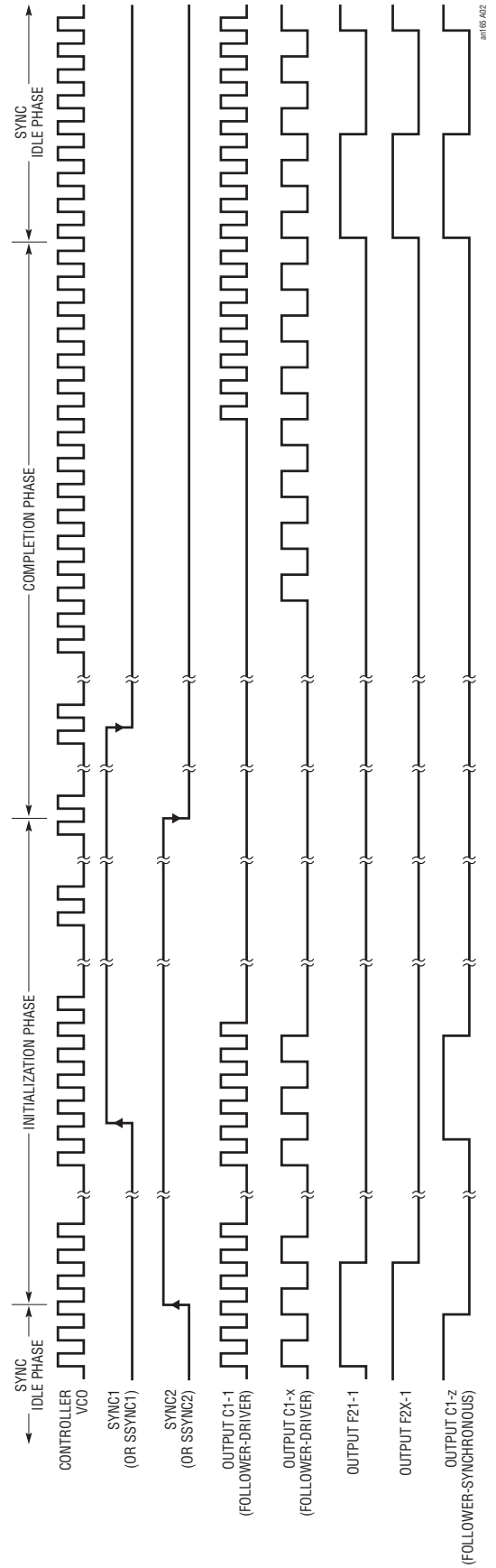


Figure A2. Sample Timing Diagram for the Figure A1 Two Stage EZSync Design. This Is Just One Simple, Illustrative Example to Show How the SYNC Signals Control the Gating on and off of the Clock Signals and How Programming the Appropriate Delay Settings Yields the Correct Output Signal Alignment of Numerous Signals. There are Hundreds of Possible Circuit Configurations and Signal Timing Combinations

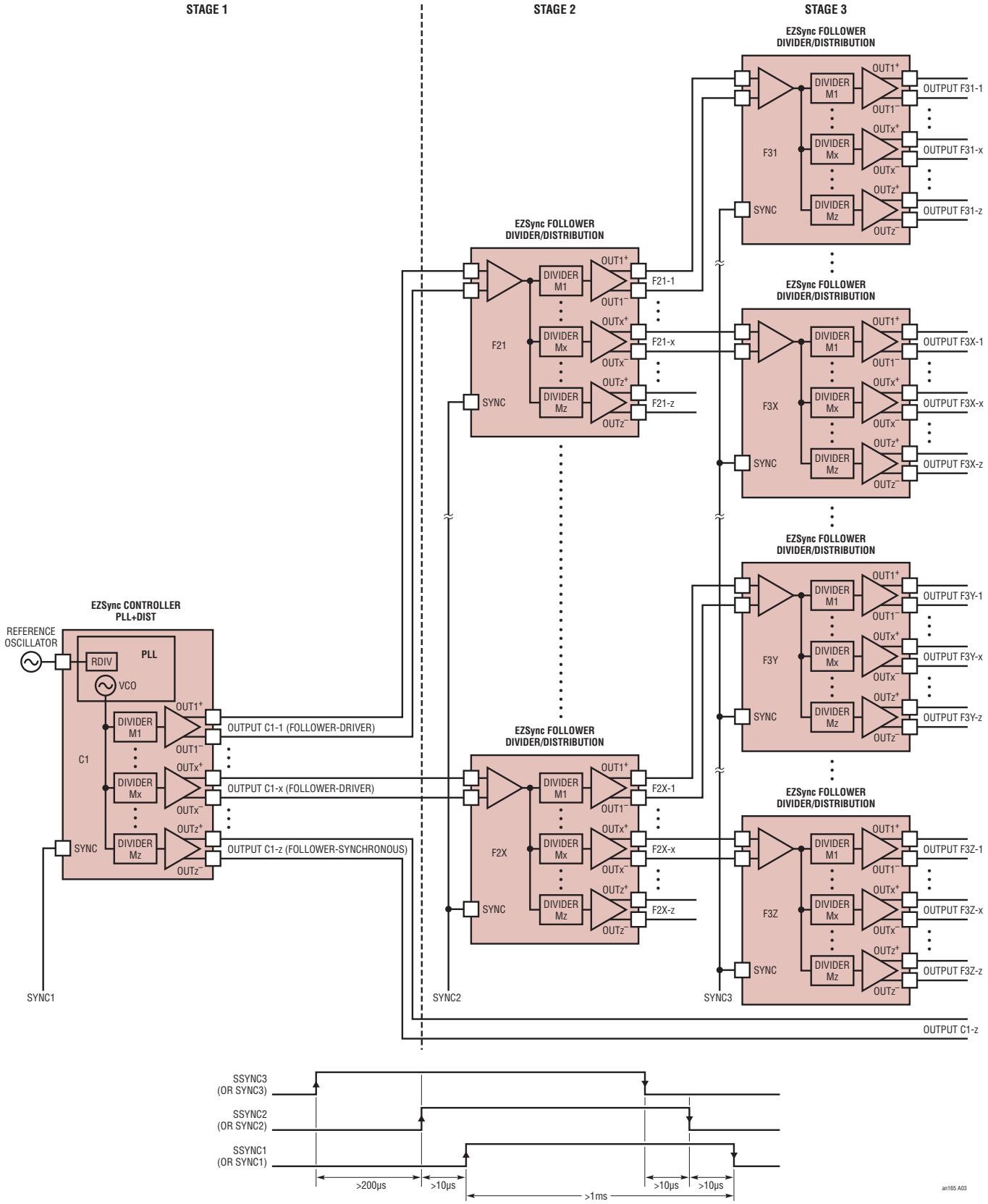


Figure A3. Sample Block Diagram for a Three Stage EZSync Design. This Is Just One Simple, Illustrative Example. There Are Thousands of Possible Configurations

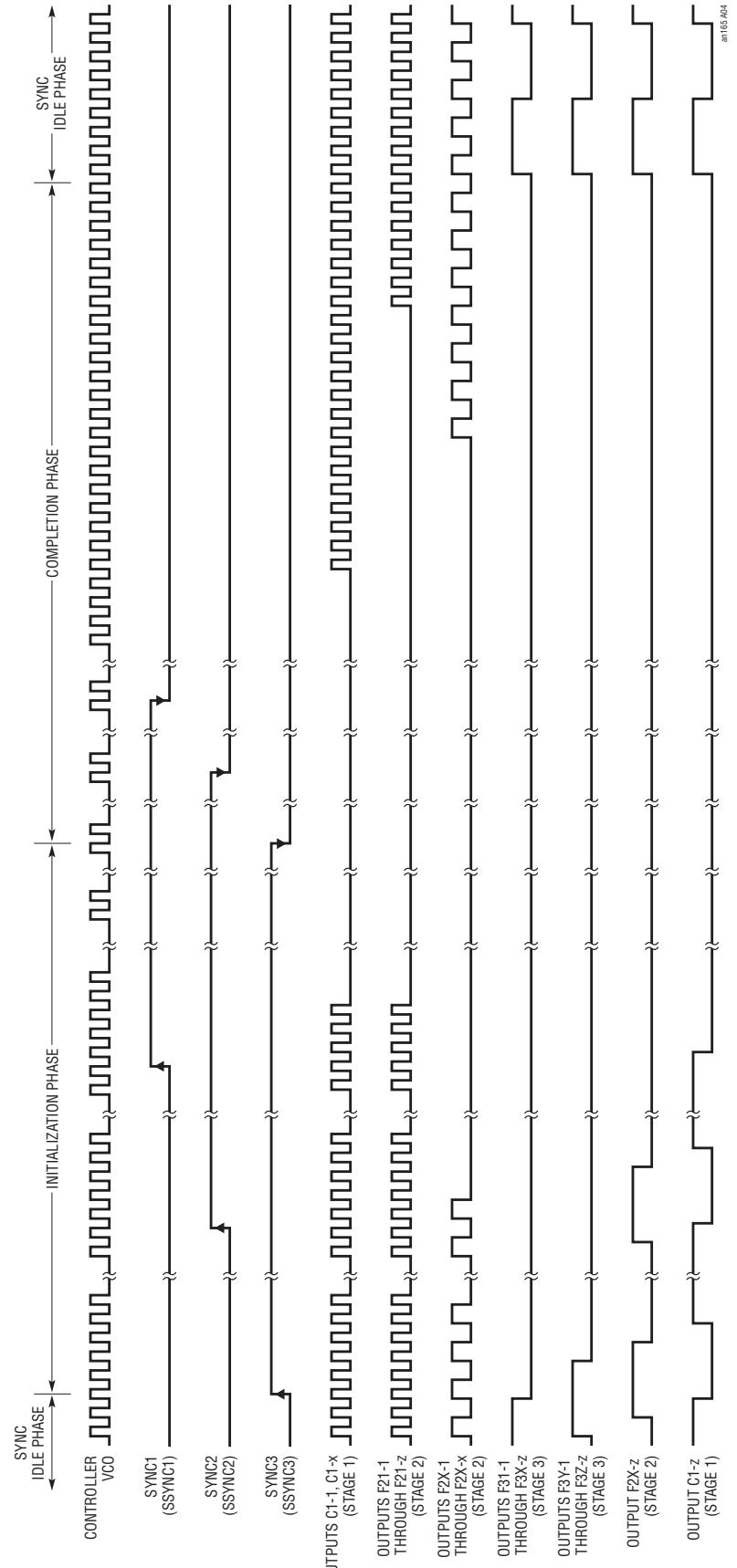


Figure A4. Sample Timing Diagram for the Figure A3 Three Stage EZSync Design. This Is Just One Simple, Illustrative Example to Show How the SYNC Signals Control the Gating on and off of the Clock Signals and How Programming the Appropriate Delay Settings Yields the Correct Output Signal Alignment of Numerous Signals. There are Thousands of Possible Circuit Configurations and Signal Timing Combinations

For applications utilizing the software controlled SSYNC register bit, the timing is fairly straightforward. All SSYNC signal transitions (rising edge or falling edge) should start at the last stage FOLLOWER level first and then continue back until the CONTROLLER is finally transitioned. For example, if there are three stages, the SSYNC signaling procedure is as follows:

INITIALIZATION PHASE

1. Write "1" to the SSYNC register bit in all of the FOLLOWER parts in Stage 3. The order of which part within Stage 3 goes first or last does not matter.
2. Wait 100µs for the synchronization circuitry on the Stage 3 parts to power up and become operational. Then wait an additional time to be certain that all of the divider outputs in Stage 3 naturally go to a logic low. By knowing the input frequency to each FOLLOWER part and the maximum divider setting for each part, the system's maximum divider time is easily calculated. In the worst case, the user should expect that each divider will need to complete one half *output* cycle plus one *input* cycle. A total wait time of 200µs is usually sufficient. This wait time may be longer than calculated if desired, but not shorter.
3. Write "1" to the SSYNC register bit in all of the FOLLOWER parts in Stage 2. The order of which part within Stage 2 goes first or last does not matter.
4. Wait for some time for the synchronization circuitry on the Stage 2 parts to power up and become operational and to be certain that all of the divider outputs in Stage 2 go to a logic low just as was done with the Stage 3 parts in step 2 above.
5. Write "1" to the SSYNC register bit in the CONTROLLER part in Stage 1.
6. Wait at least 1ms. The 1ms timing is not critical. The wait may be much longer than 1ms if desired but not shorter than 1ms.

The initialization phase is now finished.

COMPLETION PHASE

7. Write "0" to the SSYNC register bit in all of the FOLLOWER parts in Stage 3. The order of which part inside Stage 3 goes first or last does not matter.
8. Wait at least 10µs and then write "0" to the SSYNC register bit in all of the FOLLOWER parts in Stage 2. The order of which part inside Stage 2 goes first or last does not matter.
9. Wait at least 10µs and then write "0" to the SSYNC register bit in the CONTROLLER part in Stage 1.
10. Wait for a short time for all of the clock signals to propagate to the Stage 3 outputs.

The completion phase is now finished and all outputs are synchronized. The SSYNC bit must be held low.

For applications utilizing the hardware controlled SYNC pins, the timing is the same as described above for the software controlled SSYNC bit. For convenience, all of the SYNC pins at the same stage may be tied together and driven from a single logic source. So, the three stage example would require only three logic signals, one for each stage.

Last, while EZSync is very simple to use and extremely robust and repeatable, operation does rely on the SYNC pin or SSYNC bit transitions to be clean. If there are glitches or runt pulses, synchronization efforts will likely fail. This should not be difficult to achieve with modern hardware. This is potentially a larger problem with the SYNC pin than the SSYNC bit as serial port writing is very robust and the SSYNC bit may also be read back after writing to ensure its state.

Using a Divider/Distribution Part as a Pseudo-Controller

As noted earlier, *all* of Linear Technology's Divider/Distribution parts (Div/Dist) are FOLLOWER parts. However, by staggering the timing of the synchronization signals, a FOLLOWER part can perform the CONTROLLER function. When a FOLLOWER part is used in this manner we refer to it as a PSEUDO-CONTROLLER. The input clock to a PSEUDO-CONTROLLER may come from almost any source such as a laboratory signal generator, a free running, fixed frequency oscillator or an RF frequency synthesizer.

A standard CONTROLLER part has integrated timers that ensure that FOLLOWER parts completely finish each phase of the synchronization process before starting the next phase. These timers allow for a single SYNC signal to reliably synchronize two stage systems without any other assistance.

With PSEUDO-CONTROLLER operation the user *must* control the SYNC (SSYNC) signal timing to all devices and that the SYNC (SSYNC) signals *must* be staggered in time. With a PSEUDO-CONTROLLER, the use of a single SYNC signal in a two stage design is not allowed. The staggered timing of the SYNC (SSYNC) signals is similar to the three stage software synchronization (SSYNC) discussed in the previous section.

All SYNC (SSYNC) signal transitions (rising edge or falling edge) should start at the last stage FOLLOWER level first and then continue back until the PSEUDO-CONTROLLER is finally transitioned. A two stage PSEUDO_CONTROLLER block diagram with timing diagram are shown in Figure A5. The SSYNC signaling procedure is as follows:

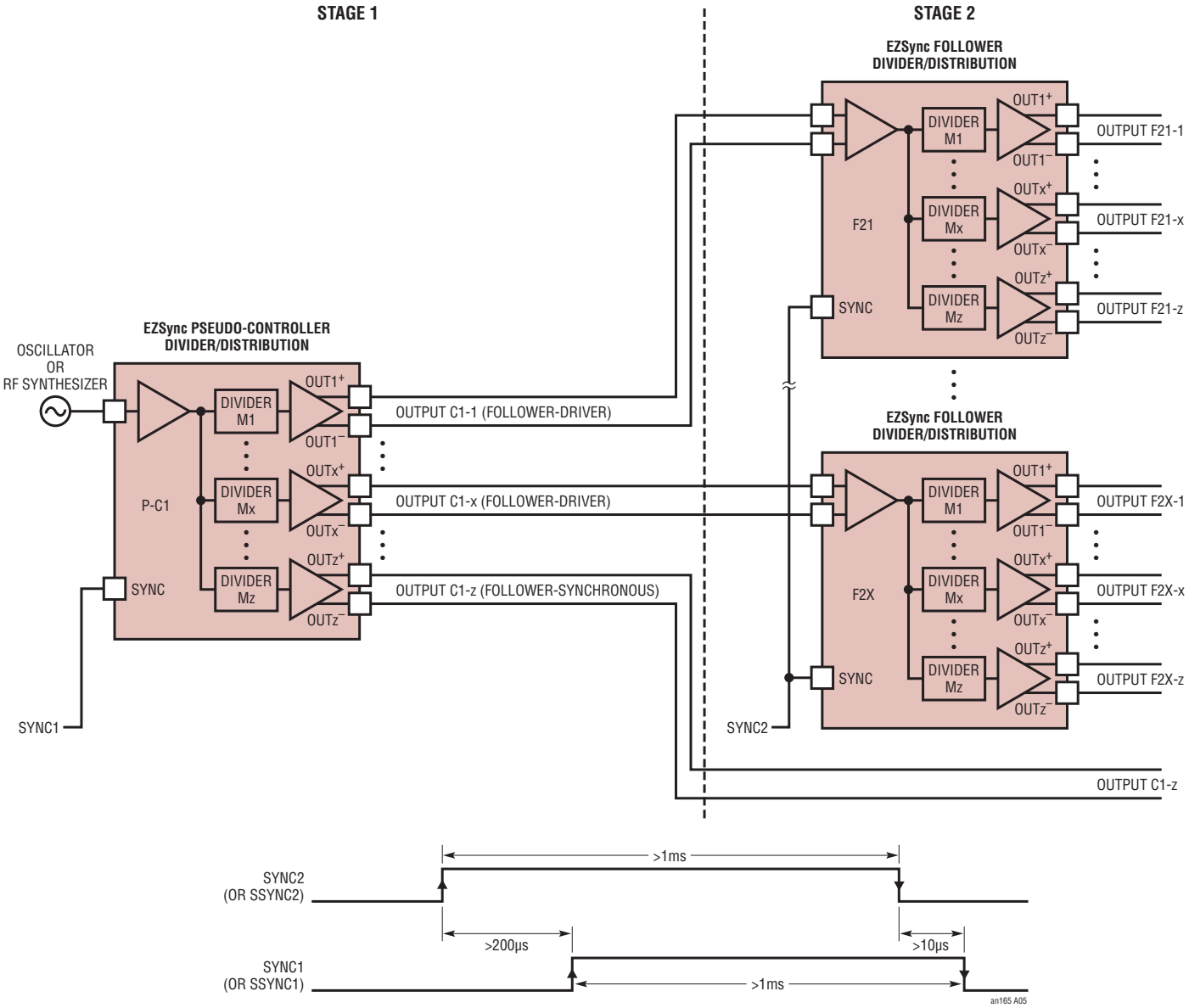


Figure A5. Sample Block Diagram for a Two Stage EZSync Design with a Standard Divider/Distribution Part Used as a PSEUDO-CONTROLLER. This Is Just One Simple, Illustrative Example. There Are Hundreds of Possible Configurations

INITIALIZATION PHASE

1. Write "1" to the SSYNC register bit in all of the FOLLOWER parts in Stage 2. The order of which part within Stage 2 goes first or last does not matter.
2. Wait 100µs for the synchronization circuitry on the Stage 2 parts to power up and become operational. Then wait an additional time to be certain that all of the divider outputs in Stage 2 naturally go to a logic low. By knowing the input frequency to each FOLLOWER part and the maximum divider setting for each part, the system's maximum divider time is easily calculated. In the worst case, the user should expect that each divider will need to complete one half *output* cycle plus one *input* cycle. A total wait time of 200µs is usually sufficient. This wait time may be longer than calculated if desired, but not shorter.
3. Write "1" to the SSYNC register bit in the Pseudo-Controller part in Stage 1.
4. Wait at least 1ms. The 1ms timing is not critical. The wait may be much longer than 1ms if desired but not shorter than 1ms.

The initialization phase is now finished.

COMPLETION PHASE

5. Write "0" to the SSYNC register bit in all of the FOLLOWER parts in Stage 2. The order of which part inside Stage 2 goes first or last does not matter.

6. Wait at least 10µs and then write "0" to the SSYNC register bit in the Pseudo-Controller part in Stage 1.

The completion phase is now finished and all outputs are synchronized The SSYNC bit must be held low.

For applications utilizing the hardware controlled SYNC pins, the timing is the same as described above for the software controlled SSYNC bit. For convenience, all of the SYNC pins at the same stage may be tied together and driven from a single logic source. So, the two stage example would require only two logic signals, one for each stage.

EZSync review

Designing a large, multipart synchronized clocking system with both low jitter and a large number of outputs is a challenge. Fortunately, Linear Technology's family of EZSync frequency synthesizers and clock distribution products addresses these concerns. The design procedure is straightforward involving only six basic steps.

1. Determine the basic design requirements:
   a. Reference and VCO Frequencies for PLL+Dist EZSync CONTROLLER or input frequency for a PSEUDO-CONTROLLER.
   b. Number of outputs and the frequency of each output.
2. Review Linear Technology's clock distribution product family and decide how to partition the design.
   a. Decide which part to use as the CONTROLLER or PSEUDO-CONTROLLER.
   b. Determine which parts will be the FOLLOWERS and how many stages are required.
3. Determine all of the divider settings for all parts to deliver the required frequency at each output.
4. Determine which outputs are synchronization disabled, Follower-Driver and Follower-Synchronous. This gives the required SYNCENx and FLDRVx bit settings.
5. For parts without the FLDRVx mode setting bit, determine the clock cycle delays for each output divider as described above.
6. Determine if the design will utilize the SYNC hardware pin or the SSYNC serial port bit for synchronization.

Steps two through six above are conceptually simple, but can be difficult to achieve in practice. There are many possible solutions each with different trade-offs. Arriving at the "best" solution can take some time.

After all of the above has been determined the detailed design process can start. PCB layout and power supply integrity are always important and the guidelines on each part's data sheet should be reviewed and followed.

APPENDIX B: PARALLELSYNC SYNCHRONIZATION IN DETAIL

Overview and Terminology

As shown in figure B1, ParallelSync is a *reference distribution* topology and is a method to synchronize multiple PLL+Dist parts running in *parallel* driven by a common reference clock *fan-out buffer network*. Synchronization is achieved through a common reference aligned signal. Synchronization through serial port commands is not an option for the ParallelSync method.

With the ParallelSync method, the system reference oscillator is fanned-out in Stage 1. In Stage 2, the PLL+Dist parts' reference divider's (RDIV) *outputs* are synchronized by a common reference aligned signal. Note that with all PLL+Dist part's RDIV outputs synchronized, all of the PLL's phase/frequency detectors (PFD) are also synchronized.

Last, recall that all of Linear Technology's PLL+Dist parts also align their outputs to the PLL feedback divider's (NDIV) output during a synchronization event. When the PLL is locked, the RDIV and NDIV outputs are aligned at the PFD (the definition of a locked PLL). Thus synchronizing the outputs to the NDIV output is the same as synchronizing to the RDIV output and the PFD. Since ParallelSync synchronizes all of the PLL+Dist parts' RDIV outputs and PFDs, all output clocks across all PLL+Dist parts are also synchronized.

PLL+Dist Part Features Required for ParallelSync Operation

Most of Linear Technology's PLL+Dist parts are suitable for use in ParallelSync applications with the exception of the LTC6950. For a PLL+Dist part to be able to support the ParallelSync method it must have the following features:

1. The part must have a synchronization *pin* and must synchronize the output dividers to the PLL's PFD either directly or through the reference divider's (RDIV) output or the PLL feedback divider's (NDIV) output.
2. The part must have a deterministically resettable PLL reference divider (RDIV).
3. Reference Aligned Output (RAO) mode is recommended in all applications and is mandatory for some PLL+Dist parts.

All Linear Technology PLL+Dist parts have a pin allocated for synchronization control. This pin is typically controlled by a simple CMOS logic signal although some parts support differential signaling. The pin is typically labeled SYNC or possibly EZS-SRQ in the case of some parts where this pin can perform multiple functions. For simplicity this pin will be referred to as SYNC here. As mentioned above, all of Linear Technology's PLL+Dist parts synchronize the output dividers to the NDIV output. Note that when the PLL is locked, the RDIV and NDIV outputs are aligned at the PFD (the definition of a locked PLL).

The PLL+Dist part must have a deterministically resettable PLL RDIV to use the ParallelSync method. Resetting all of the PLL+Dist parts' RDIVs on the same reference clock edge assures that all parts' RDIV outputs are aligned and thus all of the PFDs are aligned. This feature when combined with feature 1, ensures that the divider/driver output signals across all Stage 2 PLL+Dist parts are also aligned.

Reference Aligned Output (RAO) mode is recommended in all applications and is mandatory for some PLL+Dist parts. As shown in Figure B2, there are two methods for implementing RAO. One method is to place one of the output dividers inside the PLL as part of the feedback divider. The other RAO method adds a delay between the NDIV and the PFD inside the PLL to compensate for the output divider delay. Note that when RAO mode is disabled, the NDIV output directly drives the PFD without an additional divider or delay in the path.

The first RAO method (placing one output divider inside the PLL) is typically used with PLL+Dist parts that use a VCO prescaler frequency divider (PDIV) to lower the frequency distributed across the part and into the output dividers. With this topology, the output phase relationship of the PDIV is uncontrolled. To achieve output signals synchronized with the PFD, one of the output dividers is placed inside the PLL. When using the ParallelSync method with a PLL+Dist part that has a PDIV prescaler, the RAO mode is mandatory.

The second RAO method adds a delay between the NDIV and the PFD inside the PLL. This delay matches the output divider's delay. Figure B2 shows how a delay is added to the NDIV output to match the propagation delay of the output

dividers. With this class of PLL+Dist part the presence of the RAO mode is not absolutely necessary. However, it does give better, tighter matching of the outputs to the RDIV output. Without the RAO mode engaged, the outputs will still have a consistent *cycle* alignment to the RDIV output when the PLL is locked due to the synchronization with the PLL's NDIV output. However, the *time* alignment error will be much greater and time variation from part-to-part and over temperature will also be greater without the RAO mode engaged.

Note that ParallelSync achieves consistent and repeatable clock *cycle* alignment. Device propagation delay and the PCB trace delay are not compensated for. To achieve absolute *time* alignment of the outputs, many parts have additional *time delay* circuitry (often called analog delay or ADLY which is different from *cycle delay*) that can be used to fine tune output signal timing.

Reference Fan-Out and SYNC-RT Signal Generation

As shown in Figure B1 the reference oscillator is fanned-out using one or more fan-out buffer parts and distributed to each of the PLL+Dist part's reference input (RDIV input). The fan-out buffers must have an acceptably low additive phase noise as these signals provide the PLL's reference. The reference frequency's low offset frequency phase noise transmits directly to the output signals. All of Linear Technology's fan-out buffers have low additive noise.

Generating the reference aligned synchronization signal is a bit trickier. As labeled in Figure B1, the externally generated SYNC-EX signal is typically asynchronously produced with a random time relationship to the reference oscillator. It must then be retimed using several flip-flops to avoid metastability issues to generate the new SYNC-RT signal. The SYNC-RT signal is time aligned to the reference oscillator. This retiming may be done with individual flip-flops, or inside an FPGA and then fanned-out to all of the PLL+Dist part's SYNC pins. The SYNC-RT signal does not need to be low noise as it is only used for synchronization and does not affect the output signal's phase noise.

The propagation delay through the flip-flops, the fan-out buffers and the PCB trace routing must be considered with some care. For proper operation, the SYNC-RT signal must arrive at the PLL+Dist parts' SYNC pins meeting defined setup and hold timing requirements relative to the reference clock signal. The setup and hold specifications vary from part to part. Consult each part's data sheet for the detailed specifications.

ParallelSync Operation

At the start of a ParallelSync event all of the PLL+Dist parts must be fully configured with the PLLs locked and each part's output dividers and delays fully programmed. In this state all outputs are at the correct frequency, but have random phase relationships to each other. Additionally, each PLL+Dist part has one or more bits that must be enabled to allow resetting of the RDIV. These bits are usually called SR, SN, PARSYNC or something similar. Consult the data sheet for the PLL+Dist parts for the exact details.

In normal operation synchronization is idle. ParallelSync synchronization is a two phase process: initialization and completion. The initialization phase must be finished before starting the completion phase. Once the completion phase is finished, synchronization is again idle.

On each part, the synchronization state (idle, initialization or completion) is controlled by the signal on the PLL+Dist part's SYNC pin. With the ParallelSync method this is the retimed, reference aligned signal SYNC-RT.

In normal operation (when synchronization is idle) the SYNC-RT signal is held at a logic low state. To start the initialization phase of a ParallelSync event, the user must drive the SYNC-EX signal to a logic high state. The SYNC-EX signal is retimed by the flip-flops using the reference clock to form the SYNC-RT signal which also transitions from low to high. The SYNC-RT signal's rising edge, low to high transition, must arrive at each of the PLL+Dist parts meeting the specified setup and hold timing requirements as shown in Figure B3. This rising edge starts the initialization phase of synchronization.

As shown in Figure B3, the rising edge of SYNC-RT resets the PLL's reference divider RDIV. This abrupt resetting of the reference divider will likely cause the PLL to lose lock. Some time must be allowed for the loop to regain lock. This usually takes several loop time constants so the actual time depends on the PLL's loop bandwidth and is application dependent. For instance, if the PLL loop bandwidth is 10kHz, it could take about 500µs (five time constants) to regain lock.

Also during the initialization phase all of the PLL+Dist parts' output dividers are placed into a state where they continue

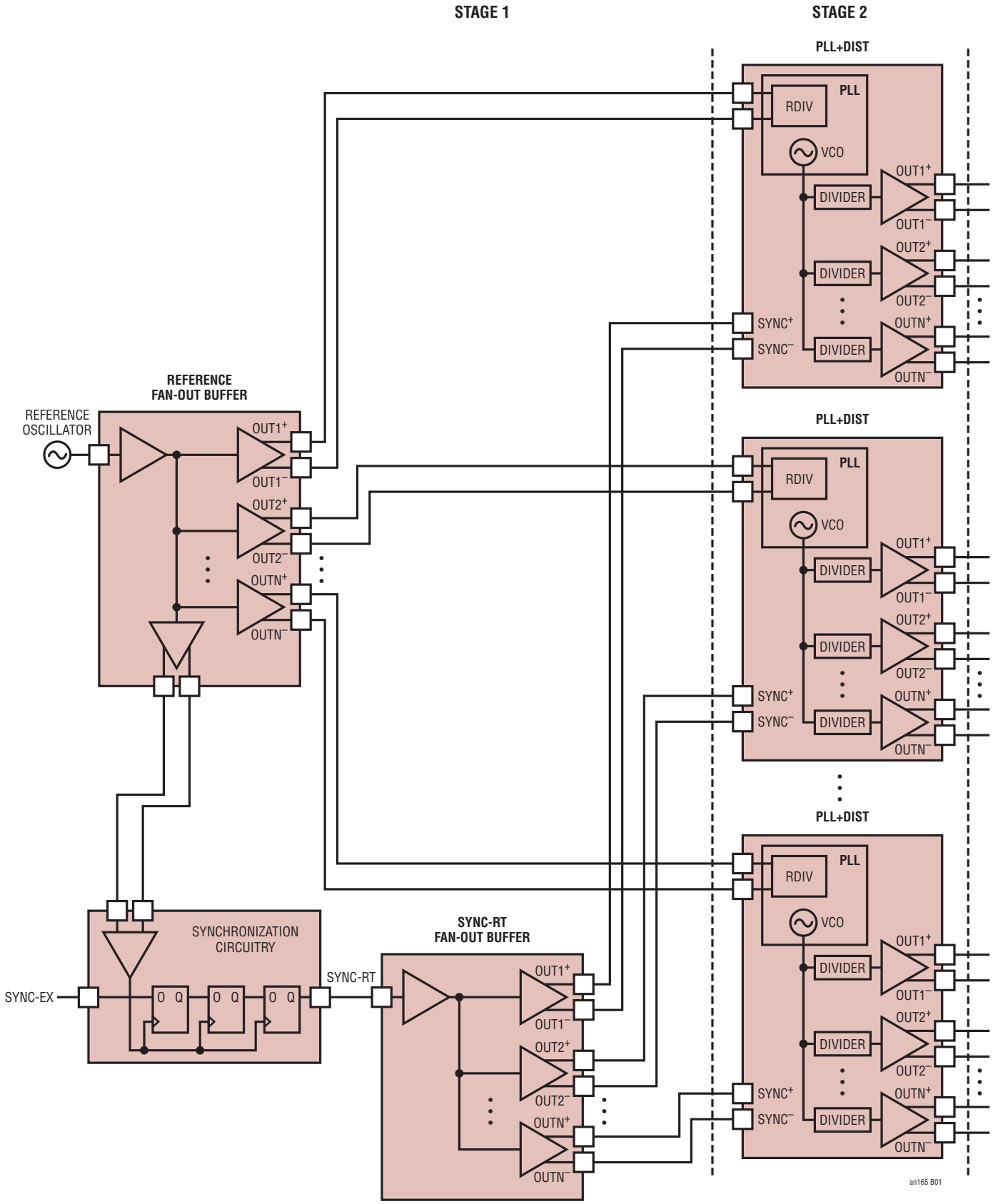


Figure B1. Block Diagram of a 2 Stage ParallelSync Design

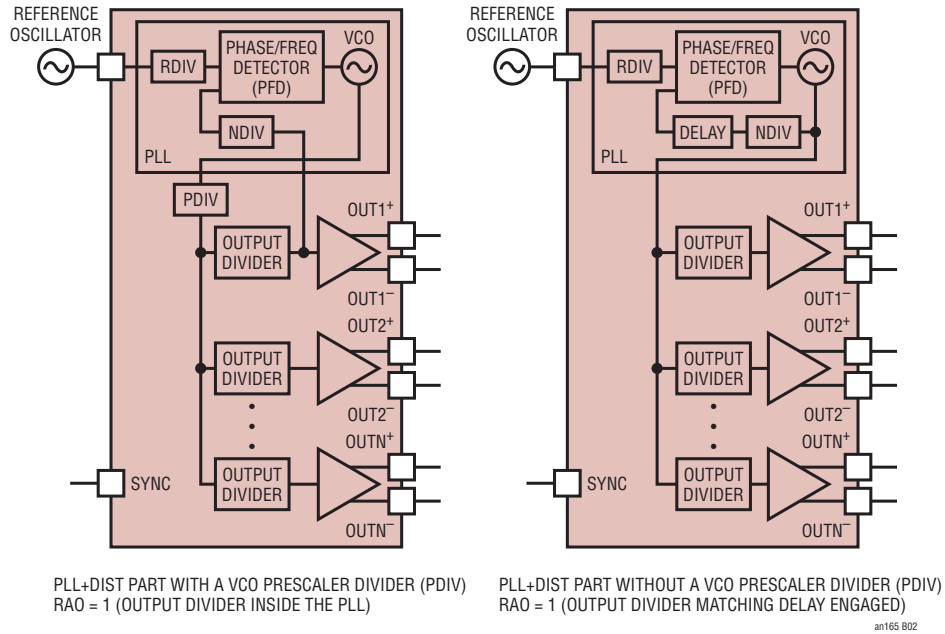


Figure B2. Block Diagram Showing Reference Aligned Output (RAO) Mode and the Two Methods for Implementing RAO

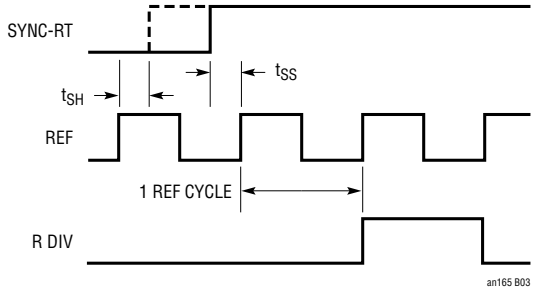


Figure B3. SYNC-RT Rising Edge Timing

to operate normally until the divider's output naturally transitions from logic high to low. Once the output goes low, it stays frozen at the low state.

SYNC-RT must be held high for the full duration of the initialization phase which must be at least 1ms. It may be held high for much more than 1ms if desired, but never less than 1ms.

At this point all RDIV outputs and all PFDs across all PLL+Dist parts are synchronized, all of the PLLs have regained lock and all outputs on all PLL+Dist parts are frozen at a logic low level. Once this state is achieved the initialization phase is finished.

To start the completion phase of a synchronization event, the user must drive the SYNC-EX signal to a logic low state causing the SYNC-RT signal to also go to a logic low after retiming by the reference oscillator. This falling edge, logic high to low transition, starts the completion phase of synchronization.

During the completion phase the SYNC-RT signal sets all PLL+Dist output dividers to be armed and ready to receive clock signals. As shown in Figure B4, after a preset number of RDIV output and VCO clock cycles, all output dividers receive their first clock edge and start normal operation. The preset number of RDIV output cycles and VCO clock cycles required varies from part to part. Consult each PLL+Dist part's data sheet or synchronization guide for the exact number cycles used by each part. A short time later all outputs from all PLL+Dist parts are delivering synchronized clock outputs.

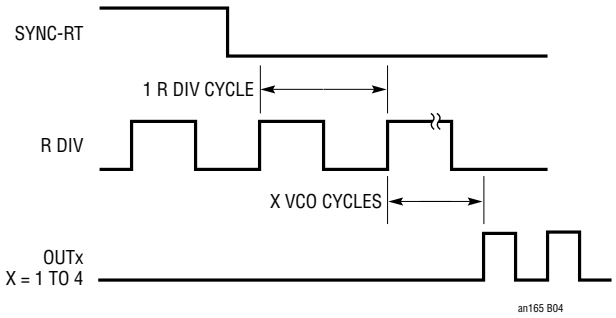


Figure B4. SYNC-RT Falling Edge Timing

At this point the completion phase is finished and the parts are all in the synchronization idle mode. The SYNC-

EX signal, and thus the SYNC-RT signal, must continue to he held low.

As noted earlier, the SYNC-RT signal must be held high for at least 1ms. In most cases the actual time that the SYNC-RT signal is held high is not important as long as it is longer than 1ms. However, should the application require that there be a repeatable, deterministic delay between the part's reference signal (the RDIV *input*) and the part's clock outputs, then the SYNC-RT logic high pulse width is important. In this case the SYNC-RT must be held high for a specific number of reference oscillator cycles (RDIV *input* cycles). The formula for calculating the minimum SYNC-RT logic high pulse width in terms of the number of reference cycles is as follows:

$$REFCYCLES = R \bullet CEILING\left(1ms \bullet \frac{f_{REF}}{R}\right) + 1$$

The minimum pulse width in time is:

$$\text{SYNC-RT Pulse Width} = \frac{REFCYCLES}{f_{REF}}$$

Where REFCYCLES is the minimum number of reference oscillator cycles that that SYNC-RT must be held high, f_{REF} is the reference oscillator frequency, R is the RDIV divider setting and the CEILING(x) function returns the smallest integer greater than or equal to x. This number is the minimum number of reference cycles that are required. If this time is too short, less than 1ms or not long enough for the PLL to regain lock, then add another R reference cycles to the REFCYCLES number calculated above.

Recall that this is optional requirement is needed only in cases where the delay between the PLL+Dist part's RDIV input and the part's outputs must be known and repeatable. Assuring a deterministic delay between the RDIV input and the output signals is a special situation and is not a requirement in most cases.

Frequency Planning Constraints

With the ParallelSync method, developing a frequency plan is quite straightforward with only one simple rule. With the ParallelSync method, all of the PLL+Dist part's phase/frequency detector (PFD) frequencies must be the same. As all of the PLL+Dist part's reference input frequencies are also the same this results in all of the PLL+Dist part's having the the same reference divider (RDIV) setting.

While this rule is all that applies for frequency planning purposes, if the VCOs in the Stage 2 PLL+Dist parts are at different frequencies, some additional part programming is required to achieve aligned outputs across all Stage 2 PLL+Dist parts. To see why, consider the timing diagram in Figure B4. The completion phase of the synchronization process requires a fixed number of VCO cycles (labeled as X cycles in Figure B4). With different VCO frequencies, the *time* for the completion phase is different for each PLL+Dist part. Thus, the outputs from different PLL+Dist parts are not aligned. However, this misalignment is consistent and may be acceptable in many cases. To correct this time offset, the appropriate number of VCO cycle delays are added to all outputs. The number of VCO cycles required for each output delay depends on the part's VCO frequency relative to the other parts' VCO frequencies. This is not difficult to determine, but must be done for proper output alignment. For assistance in setting the output cycle delays contact Linear Technology's applications department.

Expansion Beyond Two Stages

ParallelSync can expand infinitely. Stage 2 can have as many PLL+Dist devices as desired. The reference fan-out tree in Stage 1 is easy to expand indefinitely and once the SYNC-RT signal is generated it too may be fanned-out indefinitely. So, expansion to three stages or beyond is not usually needed or desirable. However, it is still certainly possible. As noted in the hybrid topologies section in the main text, the PLL+Dist parts in Stage 2 are all EZSync CONTROLLER parts and are thus capable of driving EZSync FOLLOWER parts in the next stage to give even more outputs and more output frequency options. Consult Appendix A for details about EZSync operation.

ParallelSync Review

Designing a large, multipart synchronized clocking system with both low jitter and a large number of outputs is a challenge. Fortunately, Linear Technology's family of fan-out buffers and ParallelSync capable PLL+Dist parts makes this process easier. The design procedure is straightforward involving only three basic steps.

1. Determine the basic design requirements:

a. System reference frequency and each PLL+Dist part's VCO frequency.

b. Number of outputs and the frequency for each output.

2. Review Linear Technology's clock distribution product family for fan-out buffers and PLL+Dist parts and decide how to partition the design.

   a. Determine which part or parts to use as the fan-out buffers for the reference frequency and SYNC-RT signals.

   b. Determine which PLL+Dist parts are to be used.

3. Determine how the SYNC-RT signal is generated. Discrete logic flip-flops, a shift register or using part of an FPGA are all acceptable.

Steps two and three above are conceptually simple, but can be difficult to achieve in practice. There are many possible solutions each with different trade-offs. Arriving at the "best" solution can take some time.

After all of the above has been determined the detailed design process can start. PCB layout and power supply integrity are always important and the guidelines on each part's data sheet should be reviewed and followed.

APPENDIX C: EZParallelSync AND EZ204Sync SYNCHRONIZATION IN DETAIL

Overview and Terminology

As shown in Figure C1, EZParallelSync is a *reference distribution* topology and is a method to synchronize multiple PLL+Dist parts running in *parallel* driven by a common reference clock *divider/distribution network*. Synchronization is easily achieved through a simple logic signals or SPI commands.

With the EZParallelSync method, the Stage 2 PLL+Dist parts' reference dividers (RDIV) are all programmed to one (R = 1). The RDIV frequency flexibility is essentially removed from the Stage 2 PLL+Dist parts and relocated into the Stage 1 reference Div/Dist network. The Stage 1 reference Div/Dist network must deliver synchronized signals to all of the Stage 2 PLL+Dist part's reference inputs. Note that with all PLL+Dist part's RDIVs set to one and all the RDIV inputs signals synchronized, all of the Stage 2 PLL's phase/frequency detectors (PFD) are also synchronized.

All of Linear Technology's PLL+Dist parts also align their outputs to the PLL feedback divider (NDIV) during a synchronization event. When the PLL is locked, the RDIV and NDIV outputs are aligned at the PFD (the definition of a locked PLL) and thus synchronizing the outputs to the NDIV output is the same as synchronizing to the RDIV output and, since R = 1, by extension synchronized to the reference Div/Dist network outputs.

With the reference Div/Dist network delivering synchronized signals and each PLL+Dist part delivering outputs aligned to these signals, all outputs from all PLL+Dist parts are synchronized to each other.

Stage 1 Reference Divider/Distribution Network

The reference Div/Dist network must deliver synchronized signals to all of the PLL+Dist parts' reference inputs. There are many methods to achieve this goal. In the simplest case, the reference oscillator does not need any frequency division or multiplication and can simply be expanded using simple fan-out buffers. Figure C1 shows the reference oscillator driving a Div/Dist part in Stage 1 that provides synchronized outputs at the reference frequency or at this frequency divided by an integer. All of Linear Technology's Div/Dist parts are suitable for this role.

Considered by itself, the reference Div/Dist network in Stage 1 is an EZSync network. All of Linear Technology's PLL+Dist and Div/Dist parts are EZSync capable. The reference Div/Dist network is easy to expand indefinitely using a multi-stage EZSync CONTROLLER/FOLLOWER configuration to generate a large number of frequencies or using fan-out buffers to generate multiple copies of a few frequencies. Consult Appendix A for details about EZSync operation.

Additionally, should the reference oscillator's signal have poor phase noise, an EZSync PLL+Dist part used with a

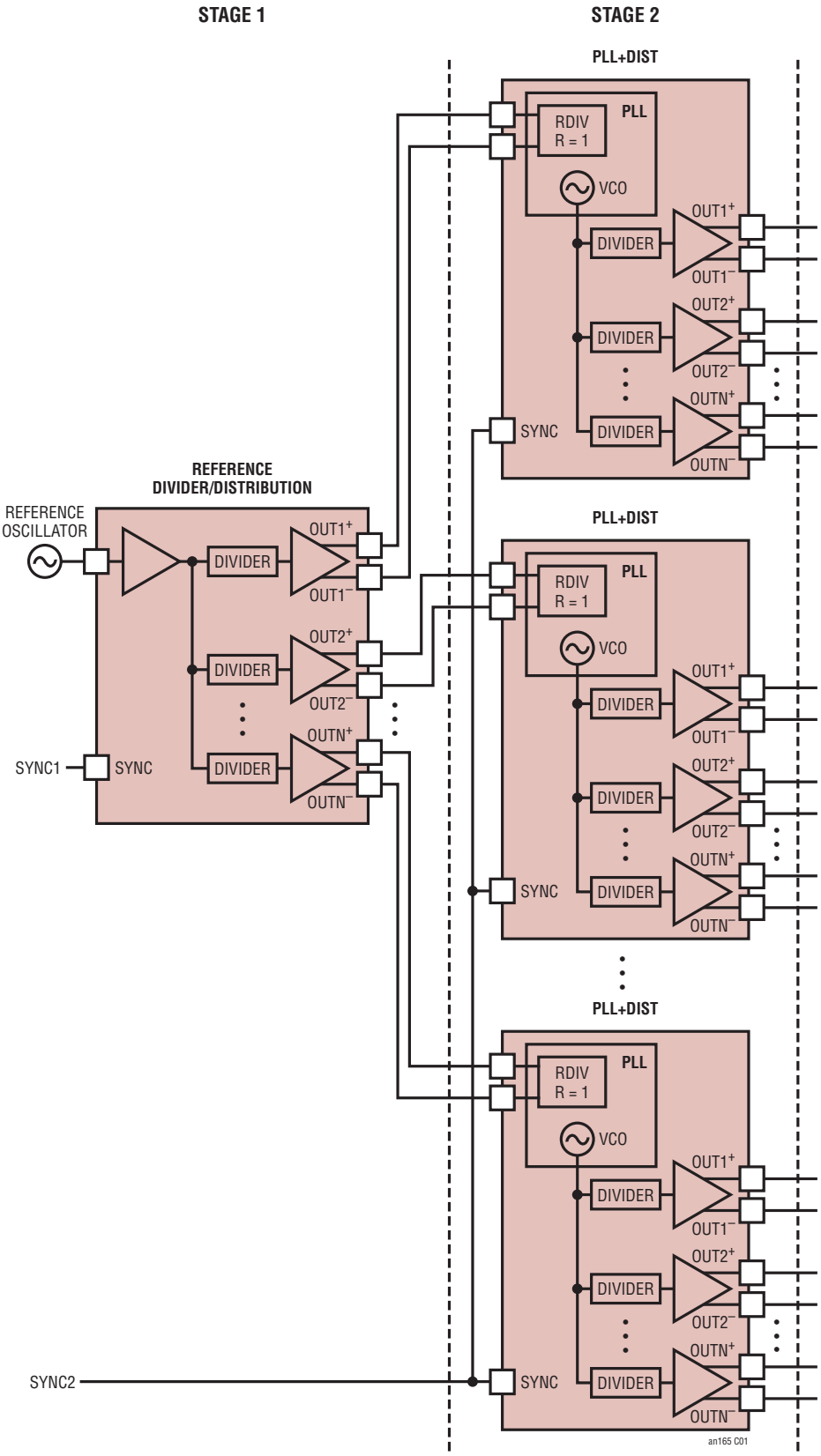


Figure C1. Block Diagram of a Two Stage EZParallelSync Design. (Synchronization May Be Done through Serial Port Commands as Well)

VCXO forms a good "jitter cleaning" function as well as delivering the synchronized signals to the PLL+Dist parts in Stage 2.

The reference Div/Dist network must also have an acceptably low additive phase noise as these signals provide the PLL's reference and low offset frequency phase noise transmits directly to the output signals. All of Linear Technology's fan-out buffers have low additive noise.

Stage 2 PLL+Dist Part Features Required for EZParallelSync Operation

All of Linear Technology's PLL+Dist parts are suitable for use in EZParallelSync applications. For a PLL+Dist part to be able to support the EZParallel-Sync method it must have the following features:

1. The part must synchronize the output dividers to the PLL reference divider's (RDIV) output or the PLL feedback divider's (NDIV) output.
2. Reference Aligned Output (RAO) mode is recommended in all applications and is mandatory for some PLL+Dist parts.
3. The reference divider (RDIV) must be set to one (R = 1).

All Linear Technology PLL+Dist parts have a pin allocated for synchronization control. This pin is typically controlled by a simple CMOS logic signal although some parts support differential signaling. The pin is typically labeled SYNC or possibly EZS-SRQ in the case of some parts where this pin can perform multiple functions. For simplicity this pin will be referred to as SYNC here.

In addition to the SYNC pin, *most* Linear Technology PLL+Dist parts also have a serial port register bit for synchronization control. The exception is the LTC6950. This register bit is simply labeled SSYNC (shorthand for Software SYNC) or possibly SSRQ in the case of some parts where this bit can perform multiple functions. For simplicity this bit will be referred to as SSYNC here.

As mentioned above, *all* of Linear Technology's PLL+Dist parts synchronize the output dividers to the NDIV output. Note that when the PLL is locked, the RDIV and NDIV outputs are aligned at the PFD (the definition of a locked PLL) and thus synchronizing the outputs to the NDIV output is the same as synchronizing to the RDIV.

Reference Aligned Output (RAO) mode is recommended in all applications and is mandatory for some PLL+Dist parts. As shown in Figure C2, there are two methods for implementing RAO. One method is to place one of the output dividers inside the PLL as part of the feedback divider. The other RAO method adds a delay between the NDIV and the PFD inside the PLL to compensate for the output

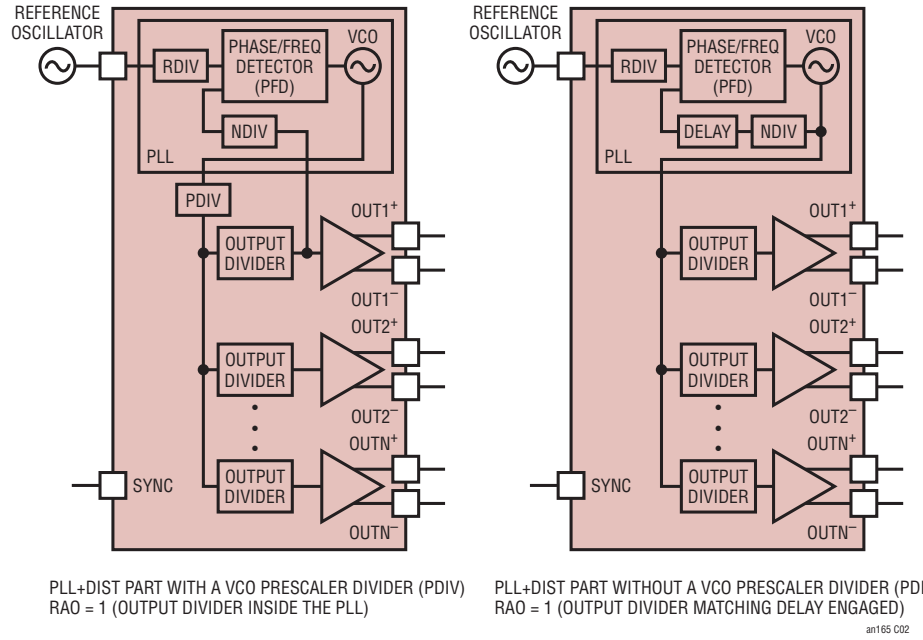


Figure C2. Block Diagram Showing Reference Aligned Output (RAO) Mode and the Two Methods for Implementing RAO

divider delay. Note that when RAO mode is disabled, the NDIV output directly drives the PFD without an additional divider or delay in the path.

The first RAO method (placing one output divider inside the PLL) is typically used with PLL+Dist parts that use a VCO prescaler frequency divider (PDIV) to lower the frequency distributed across the part and into the output dividers. With this topology, the output phase relationship of the PDIV is uncontrolled. To achieve output signals synchronized with the PFD, one of the output dividers is placed inside the PLL. When using the ParallelSync method with a PLL+Dist part that has a PDIV prescaler, the RAO mode is mandatory. Additionally, the output divider placed inside the PLL must have the lowest output frequency for the PLL+Dist part.

The second RAO method adds a delay between the NDIV and the PFD inside the PLL. This delay matches the output divider's delay. Figure C2 shows how a delay is added to the NDIV output to match the propagation delay of the output dividers. With this class of PLL+Dist part the presence of the RAO mode is not absolutely necessary. However, it does give better, tighter matching of the outputs to the RDIV output. Without the RAO mode engaged, the outputs will still have a consistent *cycle* alignment to the RDIV output when the PLL is locked due to the synchronization with the PLL's NDIV output. However, the *time* alignment error will be much greater and time variation from part-to-part and over temperature will also be greater without the RAO mode engaged.

Last, all of the PLL+Dist part must have their reference dividers (RDIVs) set to one (R = 1) to use the EZParallelSync method. PLL+Dist part features 1 and 2 above assure that the PLL+Dist outputs align to each PLL+Dist part's RDIV output and PFD. With the RDIV set to one, the outputs are thus synchronized to the PLL+Dist part's reference input as delivered from the reference Div/Dist network.

Note that EZParallelSync achieves consistent and repeatable clock *cycle* alignment. Device propagation delay and the PCB trace delay are not compensated for. To achieve absolute *time* alignment of the outputs, many parts have additional *time delay* circuitry (often called analog delay or ADLY which is different from *cycle delay*) that can be used to fine tune output signal timing. The device propagation delay and the PCB trace delay are typically consistent so the total system signal synchronization and alignment is also consistent.

Frequency Planning Constraints

With the EZParallelSync method, developing a frequency plan seems to be a bit tricky, but it is actually quite straightforward. There are two simple rules that must be followed.

First, there are constraints on the Stage 2 PLL+Dist parts' phase/frequency detector (PFD) frequency. Note that the PFD frequency of the Stage 2 PLL+Dist is the same as the RDIV output frequency and with the RIV set to one, it is the same as the Stage 1 reference Div/Dist network's output frequency.

When using the EZParallelSync method, all of the Stage 2 PLL+Dist parts' PFD frequencies must be harmonically related. Stated another way, all of the PLL+Dist parts' PFDs must be at the system's minimum PFD frequency or an integer multiple of this minimum PFD frequency. This requirement may be expressed by the following equation:

$$f_{PFD} = f_{PFD(MIN)} \bullet K$$

Where f_{PFD} is the frequency of the specific PLL+Dist part's PFD being considered, $f_{PFD(MIN)}$ is the lowest PFD frequency from all PLL+Dist parts in the system and K is an integer greater than or equal to one. Note that this requirement actually effects the usable divider settings of the Stage 1 reference Div/Dist network.

The second rule is that each of the Stage 2 PLL+Dist part's output frequencies must be integer multiples of that PLL+Dist part's PFD frequency. This requirement may be expressed by the following equation:

$$f_{OUT} = f_{PFD} \bullet L$$

Where f_{OUT} is the frequency of the specific PLL+Dist part's output being considered, f_{PFD} is the PFD frequency of the specific PLL+Dist part associated with the output being considered and L is an integer greater than or equal to one.

These two EZParallelSync frequency rules may seem to limit the usefulness of this method. However, in practice, the output frequencies that are achievable using this reference distribution topology are actually greater than with a more conventional clock tree style distribution system.

EZParallelSync Operation

At the start of an EZParallelSync event the reference Div/Dist network and all of the PLL+Dist parts must be fully configured with the PLLs locked and each part's output dividers and delays fully programmed. In this state all outputs on all parts are at the correct frequency, but have random phase relationships to each other.

In normal operation synchronization is idle. EZParallelSync synchronization is a three-step process:

1. Synchronize the Stage 1 reference Div/Dist network.
2. Wait for the PLLs in Stage 2 to regain lock.
3. Synchronize the Stage 2 PLL+Dist parts.

Synchronizing the reference Div/Dist network must be finished before starting to synchronize the Stage 2 PLL+Dist parts. Additionally, during the synchronization of the Stage 1 Div/Dist part, the output signals will stop momentarily. This will cause the PLLs of the Stage 2 PLL+Dist parts to lose lock. Some time must be allowed for the PLLs to regain lock before starting the synchronization process on the Stage 2 PLL+Dist parts. Once the Stage 2 PLL+Dist parts synchronization is finished, system synchronization is again idle.

The procedure for synchronizing the Stage 1 reference Div/Dist network and synchronizing the Stage 2 PLL+Dist parts is an EZSync synchronization event and is essentially the same both stages. The procedure to synchronize any part's outputs is a two phase process: initialization and completion.

On each PLL+Dist and Div/Dist part, the synchronization state (idle, initialization or completion) is controlled by either a dedicated pin driven by external logic or a serial port register bit controlled through a SPI write command.

All Linear Technology parts have a pin allocated for synchronization control. This pin is typically controlled by a simple CMOS logic signal although some parts support differential signaling. The pin is typically labeled SYNC or possibly EZS-SRQ in the case of some parts where this pin can perform multiple functions. For simplicity this pin will be referred to as SYNC here.

In addition to the SYNC pin, *most* Linear Technology parts also have a serial port register bit for synchronization control. The exceptions are the LTC6950 and LTC6954 parts. This register bit is simply labeled SSYNC (shorthand for Software SYNC) or possibly SSRQ in the case of some parts where this bit can perform multiple functions. For simplicity this bit will be referred to as SSYNC here.

In normal operation (when synchronization is idle) the SYNC pin and the SSYNC bit are both at a logic low ("0"). To start the initialization phase of a synchronization event, the user must either drive the SYNC pin to a logic high or write "1" to the SSYNC bit through the SPI. Note that internal to the PLL+Dist or Div/Dist part the SYNC and SSYNC signals are logically combined through an OR gate to form one signal. Each application should use either the SYNC or SSYNC signal with the other signal held to a logic low ("0") state throughout that entire synchronization process. Internal to the part, the ORed SYNC/SSYNC signal is retimed to the input signal (the VCO signal on PLL+Dist parts) through multiple flip-flops to avoid any metastability issues. On PLL+Dist parts this signal is additionally retimed to the output of the PLL's feedback divider (NDIV). This rising edge, low to high ("0" to "1"), transition starts the initialization phase of synchronization.

During the initialization phase all of the part's output dividers are placed into a state where they continue to operate normally until each divider's output naturally transitions from a logic high to low. Once the output goes low, it stays frozen at the low state and ignores the input clock signal. Once this state is achieved the initialization phase is finished. SYNC (SSYNC) must be held high for the full duration of the initialization phase which must be at least 1ms. It may be much longer if desired but cannot be less than 1ms.

To start the completion phase of the synchronization event, the user must drive the SYNC (SSYNC) signal to a logic low ("0") state using the same signal used in the initialization phase (SYNC or SSYNC with the unused signal still held low). Internal to the part, the signal is again retimed to the input signal (the VCO signal on PLL+Dist parts) through multiple flip-flops to avoid any metastability issues. On PLL+Dist parts the signal is again additionally retimed to output of the PLL feedback divider (N divider). This falling edge, high to low ("1" to "0"), transition starts the completion phase of synchronization.

During the completion phase all of the parts output dividers are first armed and ready to receive clock signals. Then

the part delivers the input clock signals (the VCO signal on PLL+Dist parts) to the output dividers. This part's outputs are now synchronized.

As stated earlier, the Stage 1 reference Div/Dist network must finish both synchronization phases before starting to synchronize the Stage 2 PLL+Dist parts. Additionally, during the synchronization of the Stage 1 Div/Dist part, the output signals will stop momentarily. This will cause the PLLs of the Stage 2 PLL+Dist parts to lose lock. Some time must be allowed for the PLLs to regain lock before starting the synchronization process on the Stage 2 PLL+Dist parts. This usually takes several loop time constants so the actual time depends on the PLL's loop bandwidth and is application dependent. For instance, if the PLL loop bandwidth is 10kHz, it could take about 500µs (five time constants) to regain lock.

The order in which the parts in Stage 2 are synchronized does not matter. This is a powerful feature as Stage 2 PLL+Dist parts may be powered down when not needed and then powered back up and resynchronized at any time. Full system clock signal alignment is still maintained since all of the Stage 2 PLL+Dist parts' outputs are aligned to the RDIV input which has not changed. Additionally, all of the powering down, powering up and resynchronizing may be done with simple SPI commands (or individual SYNC pin logic signals should the system require it). If using the SYNC logic pins, all of the Stage 2 SYNC signals may be tied together for simplicity if individual part synchronization is not required. Once the Stage 2 PLL+Dist parts synchronization is finished, system synchronization is again idle and the SYNC and SSYNC signals are held low.

Expansion Beyond Two Stages

EZParallelSync can expand infinitely. Stage 2 can have as many PLL+Dist devices as desired. The reference Div/Dist network is easy to expand indefinitely using a multi-stage EZSync CONTROLLER/FOLLOWER configuration to generate a large number of frequencies or using fan-out buffers to generate multiple copies of a few frequencies. So, expansion to three stages or beyond is not usually needed or desirable. However, it is still certainly possible. As noted in the hybrid topologies section in the main text, the PLL+Dist parts in Stage 2 are all EZSync CONTROLLER parts and are thus capable of driving EZSync FOLLOWER parts in the next stage to give even more outputs and more output frequency options. Consult Appendix A for details about EZSync operation.

The EZ204Sync Option

EZ204Sync is a subset of EZParallelSync optimized to provide the device clocks and SYSREF signals required to achieve deterministic latency with JESD204B subclass 1 data converters in a lower noise and more power efficient manner. The detailed operation of the JESD204B protocol is beyond the scope of this application note but the basics of the special clocking requirement is covered here.

JESD204B equipped data converters utilizing the subclass 1 serial link alignment feature require two clock signals. One is the device sample clock. The other is a system reference or SYSREF clock used to align the serial link for deterministic latency. The SYSREF signal must arrive at the data converter meeting defined setup and hold times relative to the device sample clock signal. With high speed data converters requiring multi-GHz device sample clocks, this is a considerable challenge. To meet this challenge it is highly desirable that both the device sample clock and the SYSREF clock come from the same source and preferably adjacent outputs from the same part. The JESD204B serial link alignment is an infrequent event so it is desirable to power down all circuitry associated with the SYSREF signal generation once the serial link is aligned.

As shown in Figure C3, with a standard EZParallelSync configuration some of the outputs from each Stage 2 PLL+Dist part are used to deliver the device sample clocks while other outputs provide the SYSREF clocks. Having both the device clock and the SYSREF clock coming as a pair from the same part is desirable as the signals will have minimal skew that tracks well over temperature. However, while the skew may be very tight at the output pins of the PLL+Dist part, it is difficult to maintain such tight skew through the PCB trace routing to the data converter. So, this initially tight skew at the PLL+Dist part may not be maintained at the data converter's sample clock and SYSREF inputs. Additionally, once the JESD204B serial link is aligned, there is no need for the SYSREF signals. It is best to power down these outputs used for SYSREF clock generation to eliminate unnecessary signal coupling concerns and to save power.

Also shown in Figure C3, the EZ204Sync approach is slightly different. With EZ204Sync, some PLL+Dist parts

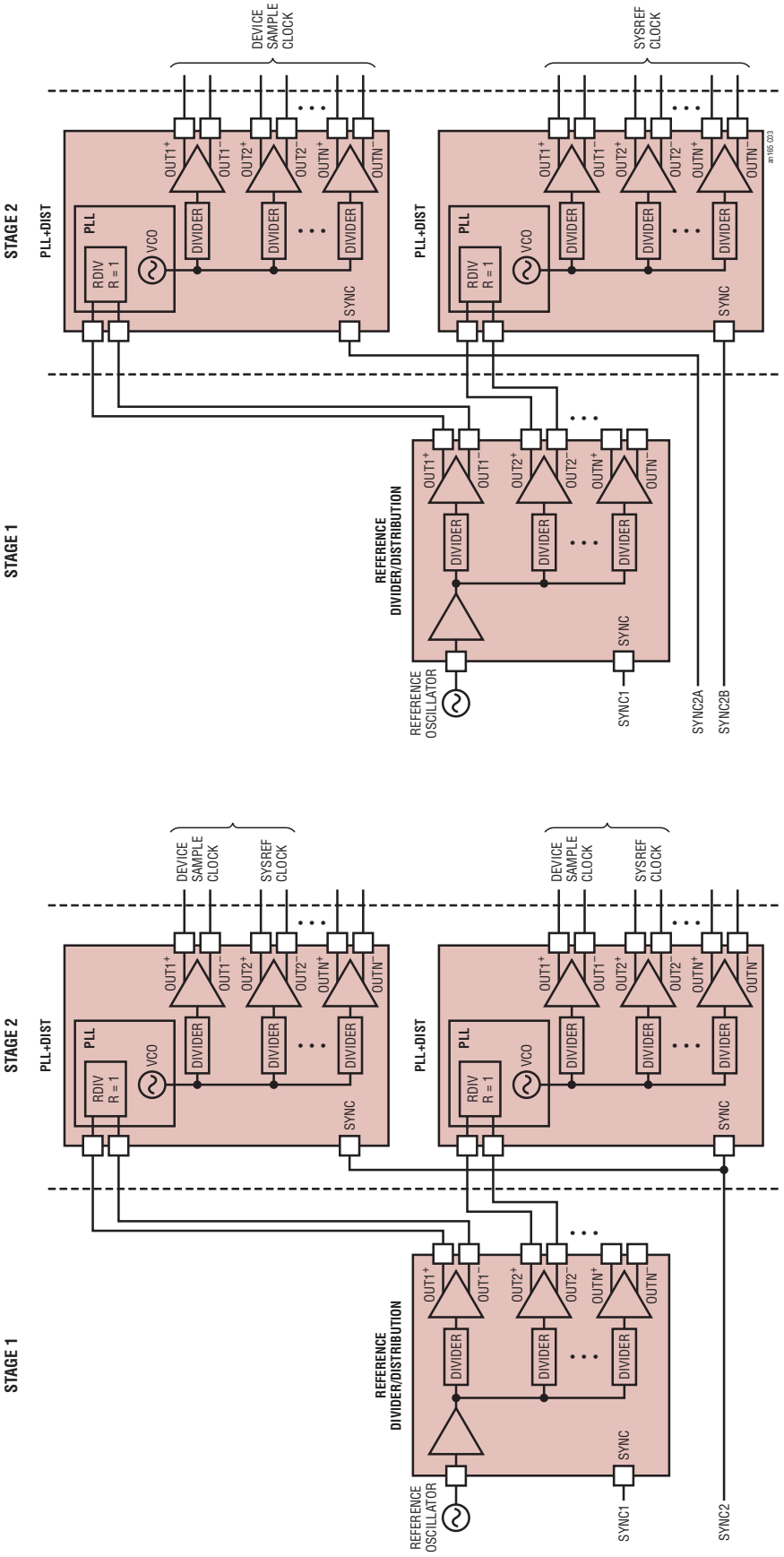


Figure C3. Comparing a Standard EZParallelSync Design with an EZ204Sync Design for Device Sample Clock and SYSREF Clock Partitioning. (Synchronization May Be Done through Serial Port Commands as Well)

Information furnished by Linear Technology Corporation is believed to be accurate and reliable. However, no responsibility is assumed for its use. Linear Technology Corporation makes no representation that the interconnection of its circuits as described herein will not infringe on existing patent rights.

are assigned to provide the device sample clocks while other PLL+Dist parts provide the SYSREF clock signals. With this approach, the device clock to SYSREF clock initial skew and temperature tracking are not as good as with EZParallelSync. However, once all PCB trace routing concerns are taken into account, this increased initial device clock to SYSREF clock skew may be less significant. Additionally, this skew may be acceptably trimmed out by using the individual time delay blocks in each of the outputs providing the SYSREF signals.

The power saving advantage of the EZ204Sync method is that, once the JESD204B serial link is aligned, the PLL+Dist part providing the SYSREF clocks may be powered down entirely saving considerable power. Additionally, if a serial link realignment is desired at a later time, only this PLL+Dist part must be powered back up and resynchronized.

In most cases, EZ204Sync also provides a lower phase noise (jitter) clocking solution than EZParallelSync. One of the constraints of the EZParallelSync and EZ204Sync methods is that the outputs of any Stage 2 PLL+Dist part must be an integer multiple of its PFD frequency. The SYSREF frequency is typically at a fairly low frequency, in the 10MHz region, while the sample clock may be several GHz. Using the EZParallelSync method may necessitate a solution where the PFD frequency is lower than optimal. The result is high PLL loop gain and thus more noise.

As a simple example, consider the systems shown in Figure C3 where the reference oscillator is 100MHz and the data converters require 1GHz device sample clocks and 7.8125MHz SYSREF clocks (1GHz divided by 128). With the EZParallelSync design, the Stage 2 PLL+Dist parts must have a PFD frequency of 7.8125MHz (or lower) and thus have a PLL loop gain of 128 to deliver the data converter's device sample clocks. With the EZ204Sync design the Stage 2 PLL+Dist parts can have different PFD frequencies. The Stage 2 PLL+Dist part delivering the data converter device sample clocks can use the 100MHz reference directly and thus have a PLL loop gain of only 10. This is 22dB less gain than the EZParallelSync design and will give much lower phase noise inside the PLL's loop bandwidth. The Stage 2 PLL part that delivers the SYSREF clocks can operate at the lower 7.8125MHz PFD frequency and will be somewhat noisier. However, the SYSREF clock is not required to be low noise as it is only used for aligning the serial data link. In this example, in addition to the power savings, the EZ204Sync design provides a lower phase noise (jitter) device sample clock than the EZParallelSync design.

EZParallelSync Review

Designing a large, multipart synchronized clocking system with both low jitter and a large number of outputs is a challenge. Fortunately, using the EZParallelSync method with Linear Technology's family of fan-out buffers, Div/Dist and PLL+Dist parts makes this process easier. The design procedure is straightforward involving only two basic steps.

1. Determine the basic design requirements:
    a. System reference frequency and each PLL+Dist part's VCO frequency.
    b. Number of outputs and the frequency for each output.
2. Review Linear Technology's clock distribution product family for fan-out buffers, Div/Dist parts and PLL+Dist parts and decide how to partition the design.
    a. Determine which part or parts to use reference Div/Dist network in Stage 1.
    b. Determine which PLL+Dist parts are to be used in Stage 2.

Step two above is conceptually simple, but can be difficult to achieve in practice. There are many possible solutions each with different trade-offs. Arriving at the "best" solution can take some time.

After all of the above has been determined the detailed design process can start. PCB layout and supply bypassing are always important and the guidelines on each part's data sheet should be reviewed and followed.

LINEAR TECHNOLOGY
NOW PART OF ANALOG DEVICES
LT 0417 • PRINTED IN USA

© LINEAR TECHNOLOGY CORPORATION 2017