Technical notes on using Analog Devices DSPs, processors and development tools
Contact our technical support at dsp.support@analog.com and at dsptools.support@analog.com
Or visit our on-line resources http://www.analog.com/ee-notes and http://www.analog.com/processors

Understanding Jitter Requirements of PLL-Based Processors

Contributed by Boris Lerner and Aaron Lowenberger

Rev 1 – January 20, 2005

Introduction

With the advance of faster processors that require faster lines of communication, understanding and characterizing clock jitter has become more important.

Jitter occurs in many different parts of digital applications. Jitter of data with respect to clock in synchronous protocols is one example; jitter of the signal itself in CDR (clock data recovery) applications is another.

This EE-Note describes jitter issues of the clock from which PLL-based processors derive timing. This document analyzes the given clock's jitter with respect to an ideal clock, only as far as a processor's tolerance requires. This document is intended for hardware design engineers responsible for choosing the components to satisfy a processor's jitter requirements.

The specific processor being analyzed in this EE-Note is the ADSP-TS201S TigerSHARC® processor. Some portions of this EE-Note apply to jitter in general; other portions apply specifically to our case in question.

Unfortunately, unlike more traditional data sheet parameters like setup and hold, analyzing acceptable system jitter is not as simple as merely ensuring that specification numbers are met. There are many ways to measure jitter; on top of that, there are infinitely many different kinds of jitter and the system may behave differently depending on the jitter type. Thus, before doing any analysis whatsoever, it is important to supply all of the jitter terms and definitions, first intuitively what they mean and then the correct mathematical definition.

Terminology

We consider an ideal clock that is being jittered (i.e., the clock's edges experience movement with respect to ideal locations).

The jitter of a particular waveform can be measured/characterized as *period*, *cycle-to-cycle*, or *time interval error (TIE)*.

Period jitter. This measures the maximum deviation of each single period of the jittered clock from that of the ideal clock. In Figure 1, if $C_0 = \text{Period of Ideal clock},$ then

$$\text{(1)} \qquad \text{Period Jitter} = \max_{k=0,1,2,3,...} \{|P_k - C_0|\}.$$

Cycle-to-cycle jitter. This measures the maximum deviation of each single period of the jittered clock from the previous period of the same clock. In Figure 1,

$$\text{(2)} \qquad \text{Cycle-to-Cycle Jitter} = \max_{k=0,1,2,3,...} \{|P_{k+1} - P_k|\}.$$

Time interval jitter. This measures the maximum deviation of the edge (Figure 1 shows this relating to the rising edge; it can also relate to the falling edge) of the jittered clock from the corresponding edge of the ideal clock. In Figure 1,

$$\text{(3)} \qquad \text{TIE Jitter} = \max_{k=0,1,2,3,..} \{|T_k|\}$$

Here we presume that at time *t=0* of the measurement, the edge of the jittered clock aligns with the edge of the ideal clock. Note that this is not just a simple aberration of the edge of the jittered clock from the nearest edge of the ideal clock, because the edge of the jittered clock may have "wandered" away from the edge of the ideal clock by more than a full cycle period.

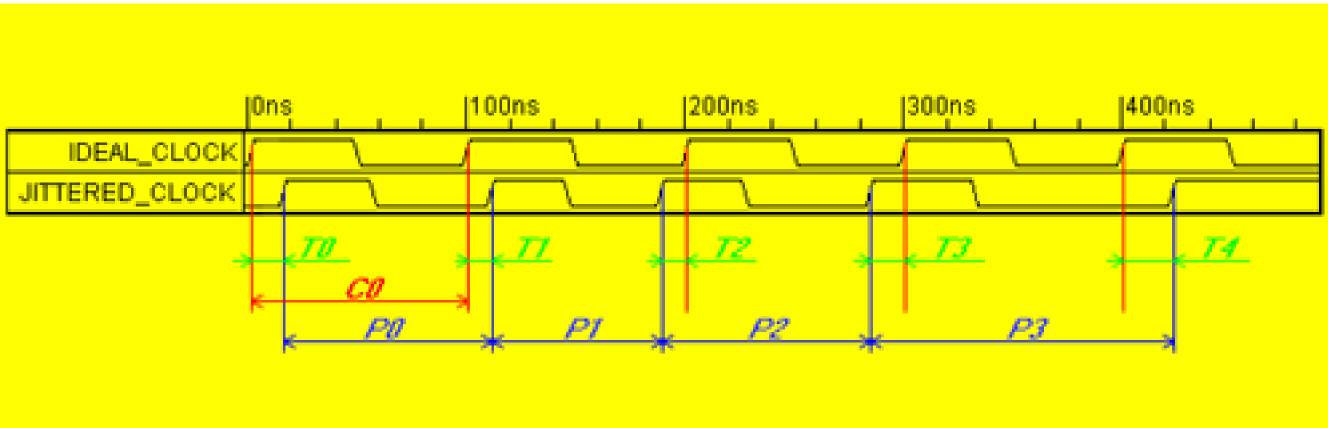

Figure 1. Jitter Definitions

It is time to put clock and jitter under a more precise mathematical definition. We define the *ideal clock of period T and constant amplitude A* as the following function of time *t*:

$$(4)\qquad C_{A,T}(t)=\begin{cases} A & \text{if } nT \le t < \left(n+\frac{1}{2}\right)T \\ 0 & \text{if } \left(n+\frac{1}{2}\right)T \le t < (n+1)T \end{cases}$$

Here, *T* is measured in units of time (usually seconds) and *A* is measured in volts. Amplitude of the ideal clock is irrelevant to the discussion about jitter, so in all that follows we presume that the clocks have unit amplitude and we consider them as parameterized by *T*, (i.e., their period):

$$(5)\qquad C_{T}(t)=\begin{cases} 1 & \text{if } nT \le t < \left(n+\frac{1}{2}\right)T \\ 0 & \text{if } \left(n+\frac{1}{2}\right)T \le t < (n+1)T \end{cases}$$

We also define $C_{J,T}(t)$, the *clock of period T with jitter J*(*t*), as the ideal clock delayed by a function of time *J*(*t*) (delay can be positive or negative), mathematically:

$$(6)\qquad C_{J,T}(t)=C_T(t-J(t))$$

There are several intuitive reasons for defining jitter this way. The most compelling reason is that jitter is usually generated in a laboratory by applying a function generator signal to a delay input of a pulse generator. This is the way processor manufacturers test susceptibility to jitter.

For simplicity we define

$$(7)\qquad G(t):=t-J(t),\ \text{so that}\ C_{J,T}(t)=C_T(G(t)).$$

Note that *G*(*t*) matters only at points

$$t=r_k\ \text{ or }\ f_k\ \text{ where } G(r_k)=kT \text{ and } G(f_k)=\left(k+\frac{1}{2}\right)T.$$

These points are precisely where the clock changes from low to high or high to low (thus "*r*" for *rising* and "*f*" for *falling* edges). Since what *G* does outside of these discrete points is completely immaterial, we can presume that *G* (and, thus *J*) are infinitely differentiable at all points.

Since these definitions are mathematical in nature and we must maintain real-life plausibility, it is safe to assume that *G* must be a uniformly increasing function (that is, $t_1 < t_2 \Rightarrow G(t_1) < G(t_2)$, since G cannot change the order of the edges, only their locations.

Going back to Figure 1, we see that $|T_k| = |J(kT)|$.

Thus, these definitions naturally lend themselves to specifying jitter as TIE (simply because the maximum amplitude of $J(t)$ is the TIE jitter, as defined in equation (3)). Unfortunately, clock driver manufacturers do not typically specify jitter as TIE. Thus, it becomes important to relate different types of jitter with mathematical formulas. This is the main reason for this application note.

Jitter Examples

Let's hesitate from this mild mathematical onslaught and examine different examples of jitter as consequences of our definitions.

Example 1

$J(t) = \varepsilon$,

where ε is a small real number. Then

$C_{J,T}(t) = C_T(t - \varepsilon)$, which is a phase

shift of the original ideal clock.

Period Jitter = 0

Cycle-to-Cycle Jitter = 0

TIE Jitter = ε

Example 2

$J(t) = \varepsilon t$,

where ε is a small real number. Then

$C_{J,T}(t) = C_T((1-\varepsilon)t)$, which is another ideal clock of

lower (if $\varepsilon > 0$) or higher (if $\varepsilon < 0$) frequency.

In fact, its period is $P_k = \frac{T}{1-\varepsilon}$, the same for all k.

$$\text{Period Jitter} = \left|\frac{T}{1-\varepsilon} - T\right| = \frac{|\varepsilon| T}{1-\varepsilon},$$

$$\text{Cycle-to-Cycle Jitter} = 0,$$

$$\text{TIE Jitter} = \infty.$$

Intuitively, function $J(t)$ is slowly increasing (or decreasing, depending on the sign of epsilon), and the edges shift uniformly. Note that infinite TIE jitter implies that this jitter cannot be generated by laboratory setup described above; it would require the function generator to output a signal of infinite amplitude. For the same reason, this type of jitter does not occur in real-life systems, so we will restrict our attention to jitter of finite amplitude.

Example 3

$J(t) = A\sin(2\pi f_0 t)$

This type of jitter is appropriately called the *sinusoidal jitter of frequency* f_0. In this case, TIE jitter (which, as noted before, is the maximum amplitude of J) is A. Deriving period and cycle-to-cycle jitter is not as trivial as in Example 1, and derivation of period jitter will be done in a following section.

Example 4

$J(t) = C_{A,T_0}(t)$, another ideal clock of period T_0 and amplitude A. It is easy to see that $J(t)$ delays the original clock by A or does not delay it at all. Thus, provided that T_0 is reasonably smaller than T,

Period Jitter = A

Cycle-to-Cycle Jitter = A

TIE Jitter = A

Example 5

$J(t)$ = White Gaussian noise. The implications of this jitter type on period, cycle-to-cycle, and TIE jitter are postponed to a latter section of this applications note, when we discuss general case jitter.

Statement of the Problem

All of the above is not intended to hopelessly confuse this topic. It is, however, intended to show the reader how confusing the subject can be - there are infinitely many types of jitter and several ways of measuring it. Let us state the problem for a designer of a clock to a processor specifically.

1. Processor manufacturers care about how short a clock pulse may become due to jitter before it violates the processor's guard-banded frequency spec. (Note that the processor's PLL usually has a jitter frequency transfer function which needs to be taken into account. It is the clock that comes out of the PLL that must abide by the frequency spec). Looking at the definitions of the jitter measurements, it is clear that the processor is concerned about the period jitter.
2. When characterizing the jitter frequency transfer function of the processor's PLL, processor manufacturers often use the lab setup described above (i.e., they really measure TIE jitter).
3. Clock driver and oscillator manufacturers do not artificially generate, but rather measure jitter and usually specify it as cycle-to-cycle.

The hardware design engineer, who is faced with three seemingly incompatible definitions, throws in a towel, goes back to school, gets an MBA and joins the marketing department instead. Or (and this happens at least as often), the engineer simply designs the board without checking the jitter specs and hopes that it works.

Jitter Frequency Domain Transfer Function Through a PLL

Without a detailed discussion of PLLs (which, in itself, has little to do with the subject at hand), we can just state that a PLL, being a phase locked loop, is linear in phase and, thus, is linear in jitter. Although this statement is not as simple as it looks, it is correct and we'll leave it at that. This is good news, because linear systems can be analyzed in terms of their frequency domain transfer function (i.e., frequency response). Note that this transfer function is linear in jitter only; a PLL itself is certainly not linear as output with respect to input. Thus, the linear system discussion that follows looks at the transfer of jitter through the PLL only.

Generally, a PLL will have a fairly flat unity gain response up to a certain frequency (because a low-frequency jitter appears as a slowly varying phase that the PLL has enough time to adjust for). Also, as is the case with all real-life systems, frequency response rolls off at the high end. The locations of these cut-off points and the response between these cut-off points varies among PLLs. The jitter transfer function for ADSP-TS201S processors with a base clock of 125 MHz and PLL multiplier set to 4 is shown in Figure 2.

Another important factor to this analysis is: given a clock at the processor maximum allowed frequency, by how much can a clock period shrink before failure. In case of ADSP-TS201S TigerSHARC processors, this value is 40 ps (i.e., 2% of the 2 ns period of a 500 MHz clock).

Armed with this data, it is time to analyze jitter tolerance. We begin with sinusoidal jitter.

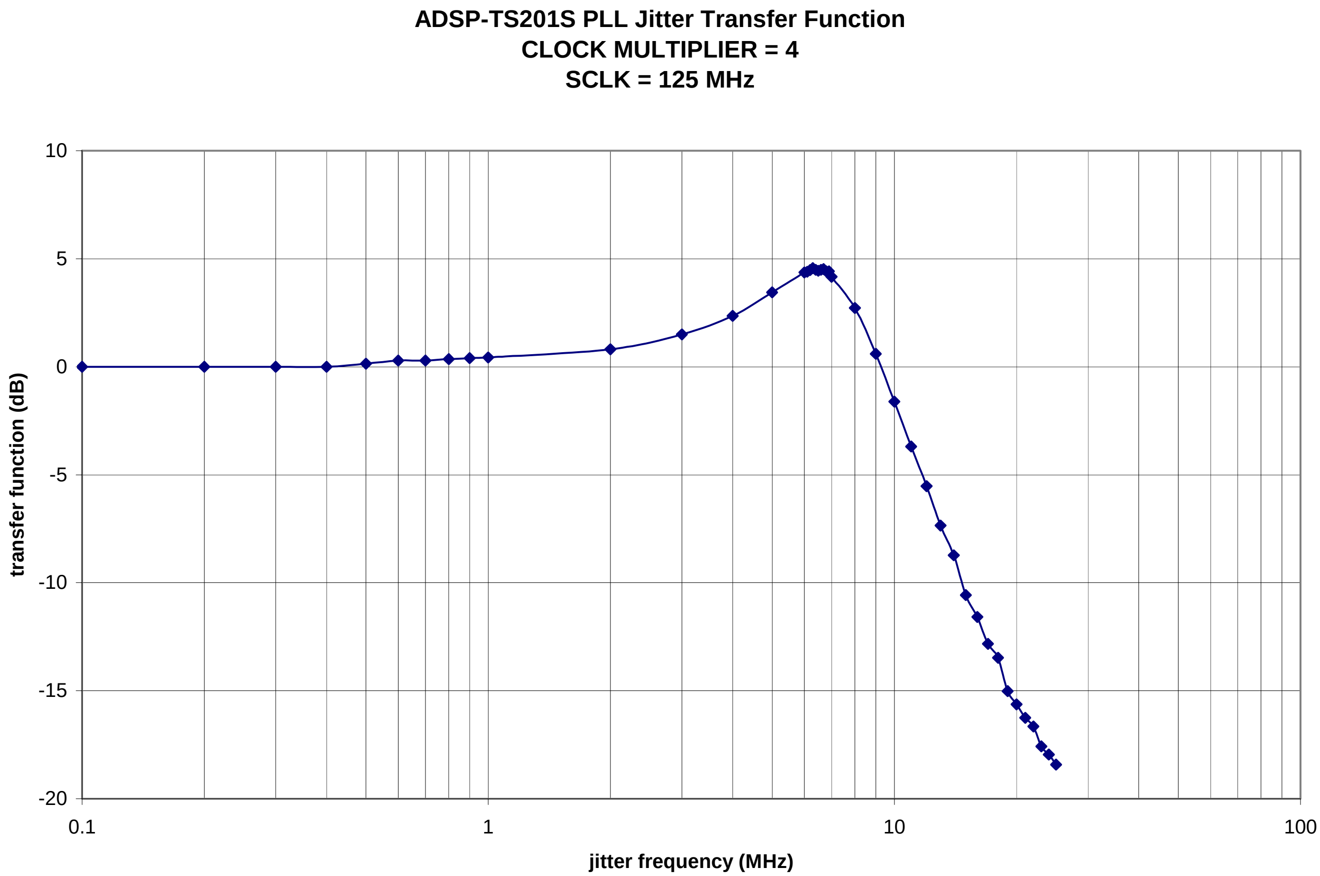

Figure 2. Jitter Transfer Function

Sinusoidal Jitter

The sinusoidal jitter of amplitude A (measured in time units, say seconds) and frequency f (in Hz) is given by $J(t) = A\sin(2\pi ft)$. From the previous discussion, we know that A is the TIE jitter.

This jitter inputs into the PLL of ADSP-TS201S and outputs $J_f(t) = A(f)\sin(2\pi ft)$, where $A(f)$ is taken from the graph in Figure 2. According to formula (1), period jitter (which, stated above, must be less than 40 ps), is given by:

$\max\limits_{k=0,1,2,3,...}\{|P_k - C_0|\}$. Since G of equation (7) is monotonically increasing, it is also invertible. The edges of $C_T(t)$ are at points kT, so the edges of $C_{J,T}(t) = C_T(G(t))$ are at points $G^{-1}(kT)$. Thus,

$$P_k = G^{-1}((k+1)T) - G^{-1}(kT)\ .$$

We now invoke the Intermediate Value Theorem from beginner calculus, which states that, given a continuously differentiable function, the difference of this function's values at the end points of an interval equals the length of that interval times the derivative of the function at some point inside the interval. We thus obtain:

$$P_k = \frac{d(G^{-1})}{dt}(t_0)((k+1)T - kT) = \frac{d(G^{-1})}{dt}(t_0)T$$

for some $t_0 \in [kT, (k+1)T]$.

Again, using beginner calculus, if

$y = G^{-1}(t)$, then $t = G(y)$ and $\frac{dy}{dt} = \frac{1}{\frac{dt}{dy}}$, i.e.

$$\frac{d(G^{-1})}{dt}(t) = \frac{1}{\frac{dG}{dt}(y)} = \frac{1}{\frac{dG}{dt}(G^{-1}(t))}.$$

Substituting this into the above (with $t = t_0$) gives:

$$P_k = \frac{T}{\frac{dG}{dt}(G^{-1}(t_0))}.$$

Since $G(t) = t - A(f)\sin(2\pi ft)$,

$$P_k = \frac{T}{1 - A(f)2\pi f\cos(2\pi fG^{-1}(t_0))} \geq \frac{T}{1 + A(f)2\pi f} \quad (8)$$

We now use Figure 2 to find the worst (i.e., maximal) value of $A(f)f$. Note that the curve in Figure 2 rolls off at the high end at the rate of 12 dB/oct. This means that every time we double the frequency, amplitude is divided by 4. Thus, $A(f)f$ is maximized at the beginning of the roll-off curve (i.e., at approximately 7 MHz) where the PLL actually boosts the jitter amplitude by about 5 dB. Thus,

$$\max_f\{A(f)f\} \leq 2^{\frac{5}{6}} A \cdot 7 \cdot 10^6 = 12.46 \cdot 10^6 \cdot A.$$

Substituting this back into equation (8) gives:

$$P_k \geq \frac{T}{1 + 2\pi \cdot 12.46 \cdot 10^6 \cdot A} \geq \frac{T}{1 + 79 \cdot 10^6 \cdot A}, \quad \text{so}$$

$$T - P_k \leq T - \frac{T}{1 + 79 \cdot 10^6 \cdot A}$$

$$= \frac{79 \cdot 10^6 \cdot A \cdot T}{1 + 79 \cdot 10^6 \cdot A} \leq 79 \cdot 10^6 \cdot A \cdot T$$

The ideal clock input in this case is 500 MHz (125 MHz input multiplied by PLL by a factor of 4; see Figure 2), so $T = 2\text{ns} = 2 \cdot 10^{-9}$ sec, so

$$T - P_k \leq 79 \cdot 10^6 \cdot A \cdot 2 \cdot 10^{-9} \leq 158 \cdot 10^{-3} A,$$

which must be $\leq 40\text{ps} = 40 \cdot 10^{-12}$ sec.

Solving this inequality for A yields

$$A \leq 0.253 \cdot 10^{-9}\ \text{sec} = 253\ \text{ps}.$$

Thus, in the case of sinusoidal jitter, to satisfy the 40 ps of period jitter required by ADSP-TS201S processors, we must ensure that the TIE of that jitter is no more than 253 ps. Unfortunately, one usually cannot assume that the jitter will be sinusoidal in nature, in which case the above analysis does not apply completely. But it does show how estimating the absolute value of the derivative of the jitter relates TIE to period jitter. Note that the above analysis can be performed more simply, without referring to the derivative. The reason that we did it in the more complicated way is because most of this argument will apply when we analyze general case jitter. We now turn our attention to this more difficult case.

General Case Jitter

As shown in the analysis above, it is very important to try to estimate the absolute value of the derivative of the jitter function on the output of the PLL.

Given jitter $J(t)$ input to the PLL, its output is $J_h(t) = J * h(t)$, where the magnitude Laplace transform of h evaluated on the unit circle

$$|H(2\pi if)| = |L(h)(2\pi if)|$$

is given by the graph of Figure 2 (as a function of f, of course).

It follows that

$$J_h'(t) = J * h'(t)$$

i.e., J_h' is the output of a filter whose impulse response is h' and input J. Since,

$$L(h')(s) = sL(h)(s),$$

the magnitude response of the filter *h'*, using the notation of the previous section is given by

(9) $\left|L(h')(2\pi if)\right| = 2\pi f\left|H(2\pi if)\right| = 2\pi fA(f)$

which is maximum at f = 7 MHz, because the high-frequency roll-off is at 12 dB/octave (this is exactly as in the previous section).

Using standard mathematical notation $\|g(t)\|_\infty$ to denote the smallest upper bound of $|g(t)|$ (i.e. pointwise amplitude of g), we now have

$$\|J_h'\|_\infty \le 2\pi\cdot 7\cdot 10^6\cdot A(7\cdot 10^6)\cdot\|J\|_\infty$$

$$\le 2\pi\cdot 7\cdot 10^6\cdot 2^{\frac{5}{6}}\cdot\|J\|_\infty \le 79\cdot 10^6\cdot\|J\|_\infty$$

Now, just like in the previous section,

$$P_k = G^{-1}((k+1)T) - G^{-1}(kT) =$$

$$\frac{d(G^{-1})}{dt}(t_0)((k+1)T - kT) = \frac{d(G^{-1})}{dt}(t_0)T =$$

$$\frac{T}{\frac{dG}{dt}(G^{-1}(t_0))} \quad \text{for some } t_0 \in [kT, (k+1)T].$$

Substituting $G(t) = t - J_h(t)$ gives

$$P_k = \frac{T}{1 - J_h'(G^{-1}(t_0))}.$$

Since $T = 2$ ns,

$$T - P_k \le T - \frac{T}{1 + \|J_h'\|_\infty} \le T - \frac{T}{1 + 79\cdot 10^6\cdot\|J\|_\infty}$$

$$= \frac{2\cdot 10^{-9}\cdot 79\cdot 10^6\cdot\|J\|_\infty}{1 + 79\cdot 10^6\cdot\|J\|_\infty} \le 158\cdot 10^{-3}\cdot\|J\|_\infty$$

To ensure that $T - P_k \le$ 40 ps, we must ensure that

$$158\cdot 10^{-3}\cdot\|J\|_\infty \le 40\cdot 10^{-12}.$$

Solving this inequality for $\|J\|_\infty$ gives

$$\|J\|_\infty \le 253\cdot 10^{-12} = 253\text{ps}$$

Thus, we end up deriving the same conclusion as was the case with sinusoidal jitter (i.e., less than 253 ps of TIE jitter ensures that the pulse jitter requirement of the processor is met).

Guidelines for Measuring TIE Jitter

Now that we know that the TIE jitter measurement is ultimately what we want, some guidelines are necessary to convert the definition of TIE jitter as given by the equation (3) into a measurement that makes sense in the real world that we live in. The problem, of course, is that the definition in (3) is theoretical in nature; the maximum is taken over an infinite number of indexes, in other words, over an infinitely long period of time. This would be rather difficult to reproduce in laboratory conditions. A greater length of time is needed to analyze a lower frequency content of jitter and, as shown by the equation (9), our TIE jitter tolerance is inversely proportional to the jitter's frequency. So, let us analyze the example at hand. Our clock frequency is 125 MHz (i.e., each cycle is 8 ns long). It is highly unlikely that this clock has more than 100 ns of TIE jitter (it would have to jitter by more than 12 periods of the clock!), if it does, there is something fundamentally wrong with the board's design. We know from the previous section that a 7 MHz jitter frequency allows about 250 ps of TIE jitter. By the inverse proportionality mentioned above, a 17.5 KHz jitter frequency should allow about 100 ns of TIE jitter (it is actually even better than that due to the jitter boost at 7 MHz). Thus, if we presume that the total TIE jitter is limited by 100 ns, all jitter content below 17.5 KHz can be ignored. Frequency of 17.5 KHz corresponds to a period of less than 60 µs (i.e., measuring TIE over a period of 60 µs should do the trick).

Cycle-to-Cycle Jitter

The previous sections relate TIE and pulse jitter. A few words need to be said in the relevance to cycle-to-cycle jitter.

The problem with cycle-to-cycle jitter is that given cycle-to-cycle jitter, no matter how small, TIE and pulse jitter can be as large as desired. A simple sinusoidal jitter of very low frequency and large amplitude will change very little from cycle to cycle (since its frequency is low), but the accumulated total change from ideal period can be very large. Thus, a board designer could not use a clock driver whose jitter is specified as cycle-to-cycle without some additional information about the nature of that jitter. In these cases, if such a driver must be used, it may be necessary to obtain a driver evaluation board and measure the TIE jitter. Then one must be careful not to assume that the jitter will measure the same on another board with a physically different driver chip. It may be necessary to contact the driver manufacturer to understand board issues that contribute to jitter (such as the power supply noise on the driver), deviation of jitter numbers across the chips, and then guard-band the measured number appropriately to ensure that the final result has no more than 250 ps of TIE jitter.

Thus, having reached this point in the application note and having survived bombardment of theoretical and practical calculations, the reader still has not been given a way to select a clock driver for our example processor! Fully realizing that, simply leaving the issue “as is”, would, most likely, also leave the unfortunate reader more than just a little peeved at the authors, we decided to contact a clock buffer manufacturer to see if they can help us resolve this dilemma. We sent the above portion of this EE-Note to the applications folks at IDT, who were more than helpful. They agreed that the cycle-to-cycle jitter specification cannot be used in processor clock designs and were happy to characterize their recommended parts jitter as TIE. For our processor example, they recommended their IDT5T9070 driver. It turns out that the jitter that this part produces depends almost entirely on the quality of its power supply. The TIE jitter on their clean evaluation board’s output measured 52 ps. At this point they were actually limited by the quality of the input clock that measured TIE jitter of 50 ps. Following this, they modulated the power supply with a 400 mV peak-to-peak wave and bypass capacitors removed (to ensure that the supply is truly modulated this much). The TIE jitter depended on the modulation frequency as shown in Table 1. Keep in mind that the frequency in Table 1. corresponds to the power supply modulation frequency. The resulting jitter frequency may be quite different.

1MHz	2MHz	3MHz	4MHz	5MHz	6MHz	7MHz	8MHz	9MHz	10MHz
314.5ps	200.2ps	163.2ps	143.6ps	140ps	127.8ps	136.1ps	148.1ps	122.6ps	125.1ps

Table 1. TIE Jitter of IDT5T9070 with Power Supply Modulated by 400mV Wave.

Note that 315 ps of jitter at 1 MHz will violate the processor’s minimum spec derived above if the jitter frequency is around 7 MHz — this is unlikely to be caused by a 1 MHz modulation of the power supply. Thus, even in the worst case of the supply modulated by 400 mV and all of the decoupling removed, the jitter spec is still met, except possibly, if the supply modulation is at 1 MHz (and even then, it is, most likely, met!).

The designer is encouraged to use the IDT5T9070 that has already been specified to meet the requirements, or, if another driver is desired, to contact the driver manufacturer directly for their TIE jitter specs.

References

[1] *ADSP-TS201 TigerSHARC Processor Data Sheet*. Rev 0, November 2004. Analog Devices, Inc.

Document History

Revision	Description
Rev 1 – January 20, 2005 by Boris Lerner and Aaron Lowenberger	Initial Release