Perceptia tm
IP and Design Services

Specifying a PLL Part 3: Jitter Budgeting for Synthesis

1 Introduction

This white paper is aimed at system architects and physical implementation leaders working on the design of SoCs.

It can be confusing to understand the impact of different jitter sources and how to calculate a jitter budget when specifying a digital system. This white paper explains how jitter changes the period of a clock and how to ensure that jitter has correctly been accounted for in the calculations for timing closure.

2 Review

This article follows on from the article "Specifying a PLL Part 2: Jitter Basics" and expands on the requirements for digital timing closure.

The following terms were defined in that article and are used again here:

Term	**Definition**
Period Jitter	Period jitter is the deviation in the period of the clock from the mean period of that clock. This is the most important jitter for digital systems.
Cycle-to-Cycle Jitter	Cycle-to-cycle jitter measures the change between two adjacent clock cycles. It does not relate to any commonly used performance criteria in digital systems and is not discussed in this article.
Accumulated jitter	Accumulated jitter, also known as long term jitter, is the deviation in the time of a given clock edge from when the same edge of an ideal clock occurs. This is not important for digital systems.
RMS Jitter	Root Mean Square (RMS) jitter can be used to quantify or specify random jitter components. The RMS value can be considered equivalent to the standard deviation (σ) of a normal distribution.
Peak-to-peak Jitter	Peak-to-peak jitter is the difference between the longest and the shortest cycle. Peak-to-peak jitter can include both random and deterministic jitter components.
Peak Jitter	Peak jitter is the difference between the shortest (or longest) cycle and the mean period of the clock. This is essentially half of the peak-to-peak jitter value.
Random Jitter	Random jitter comes from processes that are truly random such as thermal noise and flicker noise. Random jitter may also result from power supply noise, where that noise in turn originates from random processes, such as thermal noise, in the circuit supplying power to the SoC, and in particular to the PLL.
Deterministic Jitter	Deterministic jitter is jitter that follows a known pattern.

3 Jitter and Timing Closure

3.1 Timing Closure

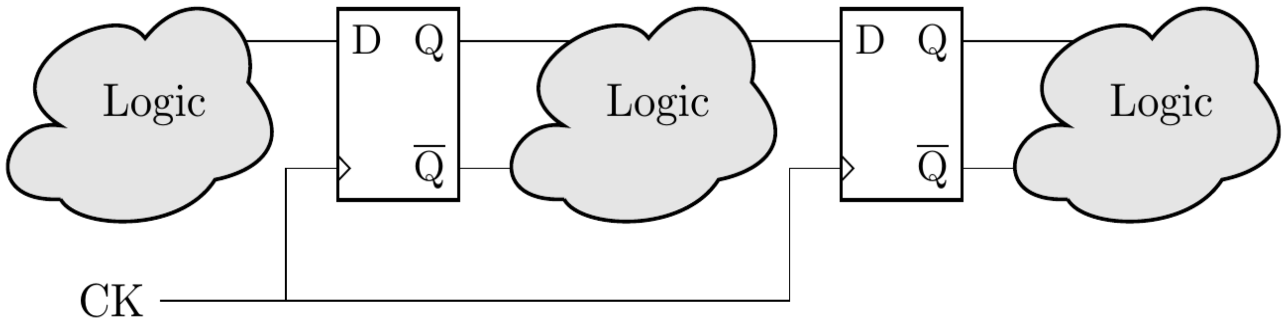

Figure 1: Typical Register to Register Logic

Figure 1 shows a typical configuration for register to register logic. Signals originate at a set of registers, are combined by a cloud of logic and are captured by another (or the same) register.

For system & physical implementation purposes, the clock cycle time defines the amount of work we can allocate to the circuits within that cycle. Controlling the logic depth and ensuring that the propagation delays fit within the defined clock period is the job of the physical implementation flow and is known as **timing closure,** which is defined by the ability of the destination registers to capture the correct result. This means that the correct result must have propagated to the input of the destination register ahead of the capture instant of that register.

Figure 2 shows an example timing diagram for the circuit of Figure 1:

- The source register, on the left of Figure 1, is clocked by the source clock on the lowest axis of Figure 2 at time t_{source} , causing a new value to be processed by the center logic cloud
- The center logic cloud has a delay $t_{propagation}$, starting from t_{source} , before generating a valid result at $t_{arrival}$
- The output of the center logic is in turn captured by the destination register on the right of Figure 1 when it is clocked by the destination clock at $t_{destination}$. The destination register will have a setup time of t_{setup} . The diagram shows a single cycle path where the capture timing is always one clock cycle later. Multi-cycle path timing calculations are analyzed in Section 3.3, following.
- The destination clock is t_{skew} earlier than the source clock, due to the difference between the two clock trees generating the clocks to the registers. (not shown on Figure 1)
- Both clocks are shown with a period of $period_{tc}$, which is the clock period budget for timing closure, defined by the primary system requirements and the capability of the technology chosen for implementation.

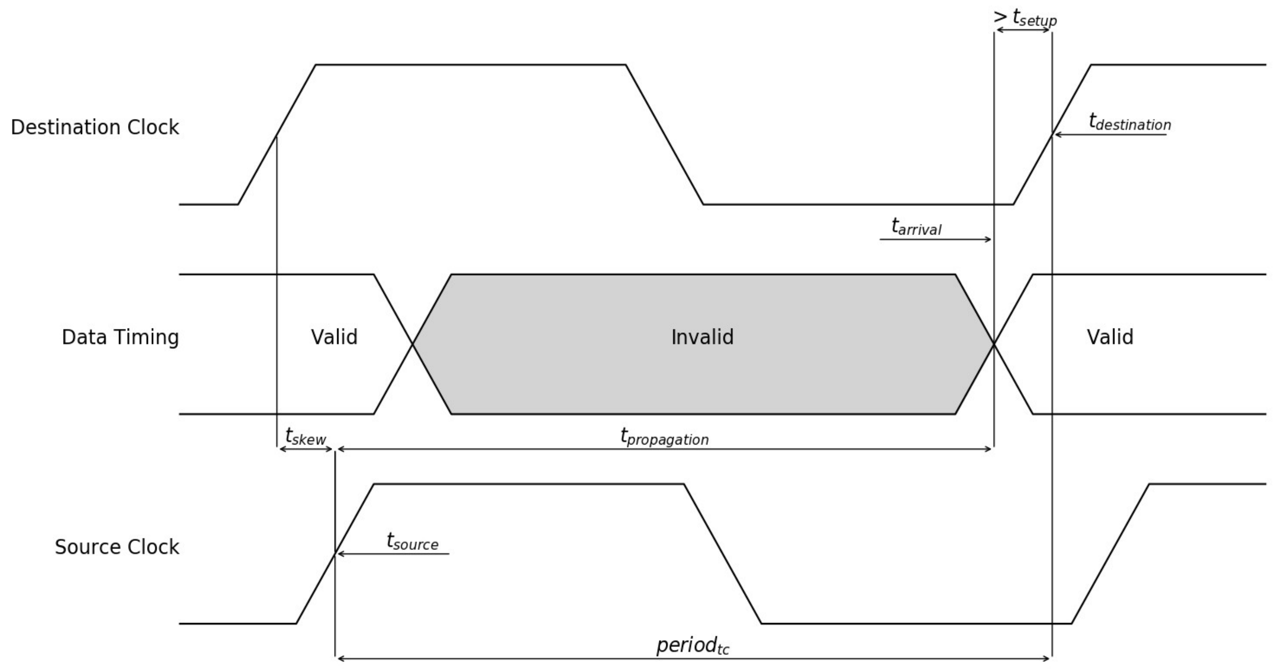

Figure 2: Example Timing

For the circuit to work correctly the input data to the destination register must be valid before it is sampled:

$$t_{arrival} < t_{capture} \tag{1}$$

Where $t_{capture}$ is the time at which the destination register captures its input.

The propagation time($t_{propagation}$) is the time taken from the clock edge causing the source register to change (t_{source}) to the arrival time at the destination register ($t_{arrival}$).

$$t_{propagation} = t_{arrival} - t_{source} \tag{2}$$

The capture time is earlier than the time the destination register is clocked by the setup time.

$$t_{capture} = t_{destination} - t_{setup} \tag{3}$$

We must also account for the skew (t_{skew}) between the time of these clocks. In addition it is good practice to add a small margin (t_{margin}) to account for errors in the calculations and factors that are difficult to account for individually.

Looking at Figure 2 we can see that the time available for a given operation can be determined from the clock period:

$$t_{destination} - t_{source} = period_{tc} - t_{skew} - t_{margin} \tag{4}$$

Putting this together we can see that the effective time available for the circuits to do their work is set by the effective clock period for timing closure ($period_{tc}$). The physical implementation flow must ensure that timing closure is met such that:

$$t_{propagation} + t_{setup} < period_{tc} \tag{5}$$

In order for the logic to be able to operate correctly, the sum of the propagation and setup times must be less than the clock period for timing closure.

Fortunately, the tools for the physical implementation are able to track the propagation, setup and skew times associated with every path in the design. This means that we do not need to calculate this every time. These tools only need the correct information on the period, and margin to ensure timing closure for the whole design by modifying the circuits to control $t_{propagation}$ and t_{skew} .

Hold times are not considered in this article as jitter is not a factor in their calculation.

3.2 Jitter

As described in “Specifying a PLL Part 2: Jitter Basics”, it is critical to know the **period** jitter of a clock source in order to achieve timing closure as this can change the amount of time available to complete a unit of work.

As we have shown above, the primary requirement for timing closure constraint development and analysis is to know the **period** of a clock source as this materially affects the amount of time available to complete a unit of work. In the article, “Specifying a PLL Part 2: Jitter Basics” we showed that the most important type of jitter to consider in this situation is period jitter as this directly reduces the effective period of the clock.

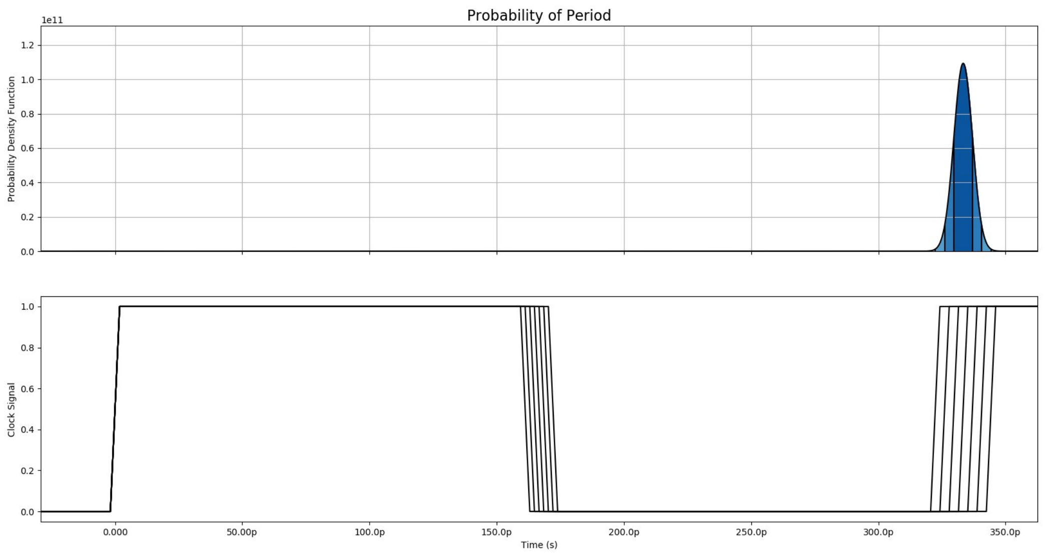

Figure 3: Period Jitter

Figure 3 reproduces the figure from the jitter basics article showing that there will be some variance in the period of any clock in the real world.

Variances that change the period of the current cycle, compared to the previous cycle (cycle-to-cycle jitter) and the variance in timing between one cycle's period and another cycle's period many thousands of cycles later (long term jitter) is not relevant in digital clocking systems and is not considered in this article.

Thus the following analysis will focus solely on **period** jitter calculations.

The system and physical implementation timing budgets now need to account for the various timing uncertainties by ensuring that $period_{tc}$ is always long enough to ensure timing closure as described above. For this to be true:

$$period_{tc} \leq period_{spec} - j_{CK} - j_{CT} \qquad (6)$$

Where:

- $period_{tc}$ is the clock period budget for timing closure, defined in Section 3.1
- $period_{spec}$ is the specified(ideal or target) clock period
- j_{CK} is the peak clock source jitter
- j_{CT} is the peak clock tree jitter

In the simplest terms, this means that $period_{tc}$ must be shorter than the shortest clock period expected for a given amount of jitter.

The following sections examine how to specify and budget for the clock jitter associated with a PLL clock source (though it may also be a useful aid for other types of clock sources).

3.3 Jitter for Multi-Cycle Paths

In some designs, it is not possible to constrain a path so that $t_{propagation}$ less than than $period_{tc}$. In these circumstances, it is possible to add special circuitry to only sample the data when it is valid and timing closure can be achieved across multiple periods of the clock. This is called a multi-cycle path and the logic is constrained so that:

$$t_{propagation} + t_{setup} < period_{tc} \times N \qquad (7)$$

For a predetermined number (N) of clock cycles.

When multi-cycle paths are used, the jitter across all each of the clock cycles that are used to make the calculation must be added to get the overall jitter across the total evaluation time. In many circumstances, there is a strong correlation between the jitter of nearby cycles, so the overall jitter can be near to worst case for several cycles in a row. For example, jitter from low frequency modulations, like that resulting from the ripples from a switched mode power supply output, will have almost the same effect for multiple cycles.

As a result of this, the only safe way to account for jitter over multi-cycle paths, is to assume the jitter is the same on all cycles use in the multi-path and:

$$N \times period_{tc} \leq N \times period_{spec} - N \times j_{CK} - N \times j_{CT} \qquad (8)$$

So the jitter (in seconds) will be increased by multiplying it by the number of cycles used in the multi cycle path. This will be the same percentage of the multi-cycle path as it was of the simple single clock cycle path.

4 Calculating Jitter Requirements

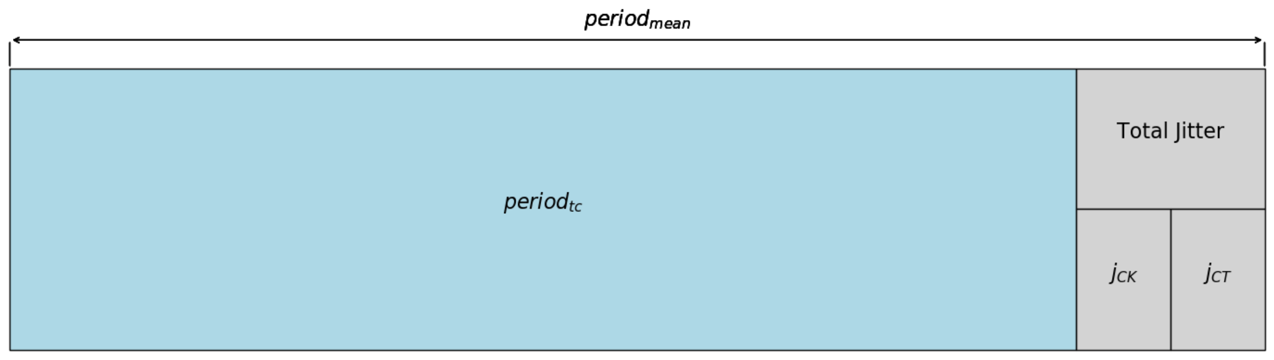

Figure 4: Clock Budgeting

To calculate your jitter requirements, follow these steps:

1. Determine the peak period jitter budget
   - Typically this is 5% of the period for moderate digital clocking applications, but may be lower for high performance applications where you need to squeeze more work out of each clock cycle.
   - In systems that can operate at multiple frequencies the jitter budget should be considered at all operating frequencies although the highest frequency is almost always the worst case.
2. Find the parameters describing the jitter for a given clock source. For a PLL, this will be the sum of the following parameters, all of which should be found on the PLL datasheet
   - The base period jitter of the PLL
   - Period jitter due to power supply noise. This power supply noise can include:
     - Switched mode power supply ripple
     - Noise from activity of other circuits sharing the supply
     - Noise from other PLLs sharing the supply
3. Calculate the individual jitter components from the jitter parameters and system conditions
4. Convert all the jitter components to peak period jitter
   - Perceptia specifies peak period jitter on our datasheets, so you have the information you require directly
   - If you have peak-to-peak period jitter, then divide by 2 to get peak period jitter
   - If you have RMS period jitter multiply by 5 to get peak period jitter. Other multiples may be used here in place of 5, as described in Section 5.1 of the article "Specifying a PLL Part 2: Jitter Basics"
5. Calculate the total jitter of the clocking source by adding all components.
6. Calculate the jitter from any other sources
   - This can include clock tree jitter

NOTE: clock tree jitter is primarily due to power supply variation at the local clock tree buffers. Since CMOS logic delays are directly proportional to the supply, variance or noise on the power supply to the clock tree gates will cause variance in the root to leaf delay of the clock tree. This is generally accounted for within the physical implementation tools and lumped into the leaf-to-leaf skew margin number generated internally by the clock tree synthesis tools.

Examples are given in Section 7.

5 Datasheet Jitter Parameters

A typical datasheet will give you a number of parameters from which the total jitter can be calculated. The exact parameters will depend on the particular supplier.

Perceptia specifies the following parameters which can be used to calculate the jitter:

Parameter Name	Description	Typical Value	Units
K_V	Power supply noise parameter: Multiply this by the power supply noise to get the jitter component due to power supply noise. All power supply noise sources should be added, including switch mode power supply noise and noise from other blocks that share the supply.	0.05	%/mV
K_{CC}	PLL cross coupling parameter: This parameter is used to calculate the impact on the jitter of sharing a single supply across multiple PLLs. This parameter is multiplied by the effective power supply resistance, which takes into account the number of PLLs that share specific parts of a supply network.	0.004 to 0.012	%/Ω/PLL

These are parameters that the designer of the designer of a digital system has control over. The designer may tune K_V by choosing a different switched mode power supply or adjusting the amount of logic sharing a supply with the PLL. The designer also has the ability to tune K_{CC} by changing the clock skew between the inputs to different PLLs that share a single reference clock.

The examples below demonstrate how changing these parameters can have a significant impact on the jitter of a system.

6 Improving Jitter

There are a number of ways to improve the jitter if a given setup does not meet you requirements:

- Switch to a higher performance PLL
  - This will improve the base jitter and may also change the sensitivity to other factors.
  - The jitter performance of a PLL may need to be traded off against cost, power and/or area.
- Reducing power supply noise This can be done by:
  - Improving the power supply impedance
  - Using a regulator with lower supply noise, this can be a switched mode power supply with lower ripple or a linear regulator, depending on the requirement.
  - Changing which other elements share power supplies with the PLL

See the examples below for an indication of the effects of these strategies.

7 Jitter Calculation Examples

The table below shows four example calculations of the jitter:

- The single PLL case is an example case for a single PLL with an adequate power supply network.
- The Dual PLL case is an example of two PLLs sharing a single supply network with an adequate power supply network.

- 8 PLL case 1 is an example of 8 PLLs sharing a single supply network with a poor power supply network, and significant power supply noise.
- 8 PLL case 2 is an example of 8 PLLs sharing a single supply network with a adequate power supply network.

Calculation Details	**Single pPLL03 case**	**Single pPLL02 case**	**8 pPLL03 case 1**	**8 pPLL03 case 2**
PLL Used	pPLL03	pPLL02	pPLL03	pPLL03
N_{PLL}	1	2	8	8
$(R_{V\,EFF}+R_{G\,EFF})$: Effective power supply and ground resistance calculated from realistic supply networks.	4	4	13.4	8
J_{base} **for PLLs from datasheet**	**1.2%**	**2.1%**	**1.2%**	**1.2%**
V_{noise}	10mV	10mV	25mV	10mV
K_V from PLL datasheet	0.05	0.05	0.05	0.05
$J_{noise}=V_{noise}*K_V$	**0.5%**	**0.5%**	**1.25%**	**0.5%**
K_{CC} : Common CK_REF with no clock skew from PLL datasheet	0.012	0.012	0.012	
K_{CC} Common CK_REF with significant clock skew between different PLLs, from the datasheet				0.004
$J_{CC}=(R_{V\,EFF}+R_{G\,EFF})\times K_{CC}$	**0.05%**	**0.05%**	**0.16%**	**0.12%**
$J_{total}=J_{base}+J_{noise}+J_{RN}+J_{CC}$	**1.75%**	**2.65%**	**2.61%**	**1.82%**
Percentage of the clock available for work	**98.25%**	**97.35%**	**97.39%**	**98.18%**
Time available for work on a 3GHz clock	**327.5ps**	**324.5ps**	**324.6ps**	**327.2ps**

8 Conclusion

Clock jitter is a key parameter for digital designs and all designers should have an understanding of its effect on timing closure and the measures available to them in order to control it.

In the overall jitter of any system can be set by the correct choice of PLL and the correct setup of that PLL. This article has attempted to give designers a better insight into how to best achieve this.

9 About the Author

Julian Jenkins, is the CEO and CTO of Perceptia Devices. As CTO, he is the architect of all of Perceptia's PLL IP where he been among the pioneers in the development of all digital PLLs. Julian has several US and international patents as a result of this work. He has participated in an impressive list of high-speed analog and mixed-signal ICs that are in commercial production. Feel free to contact Julian if you have questions about this topic.

Perceptia Devices is an IP and design services provider, based in Sydney, Australia and Silicon Valley. It is focused on PLLs for RF systems and other demanding clocking applications.