Clock Tree Design Considerations

Hardware design in high performance applications such as communications, wireless infrastructure, servers, broadcast video and test and measurement is becoming increasingly complex as systems integrate more functionality and require ever-increasing levels of performance.

Clock Tree Design Considerations

Hardware design in high performance applications such as communications, wireless infrastructure, servers, broadcast video and test and measurement is becoming increasingly complex as systems integrate more functionality and require ever-increasing levels of performance. This trend extends to the board-level clock tree that provides reference timing for the system. No “one size fits all” strategy applies when it comes to clock tree design. Optimizing the clock tree to meet both performance and cost requirements depends on a number of factors, including the system architecture, IC timing requirements (frequencies, signal formats, etc.) and the jitter requirements of the end application.

Reference Timing – When to Use a Crystal vs. a Clock

One of the first design considerations is to inventory the hardware design’s reference clock requirements and select the type of reference clock that will be used for the processors, FPGAs, ASICs, PHYs, DSPs and the various other components in the system. Quartz crystals are typically used if the IC has an integrated oscillator and on-chip Phase-Locked Loops (PLLs) for internal timing. Crystals are low cost components, exhibit excellent phase noise and are widely available. They can also be placed in close proximity to the IC, simplifying board layout. One of the drawbacks of crystals, however, is their frequency can vary significantly over temperature, exceeding the parts-per-
millions (ppm) stability requirements of many SerDes applications. In many stability-sensitive high-speed SerDes applications, crystal oscillators (XOs) are recommended because they guarantee tighter stability than passive crystals. Often, XOs are used for SoC reference timing when one or two reference clocks are required.

Clock generators/buffers are more typically used when several reference frequencies are required. In some applications, FPGA/ASICs have multiple time domains for data path, control plane and the memory controller interface and require multiple unique reference frequencies. A clock generator/buffer is also preferred when the IC cannot accommodate a crystal input, the IC must be synchronized to an external reference (source-synchronous application) or a high frequency reference not easily generated by a crystal is necessary.

Free-Running vs. Synchronous Clock Trees

Once the hardware design clock inventory is completed and crystals have been selected for some of the components, the next step is to select the timing architecture for the remaining clocks: free-running or synchronous. For applications that require one or more independent reference clocks without any special phase-lock or synchronization requirements, XOs, clock generators and clock buffers are the preferred choice. Processors, memory controllers, SoCs, and peripheral components (e.g. USB, PCI Express switches) typically use a combination of XOs, clock generators and clock buffers for reference timing in free-running, asynchronous applications. XOs are preferable when the application requires 1 to 2 timing sources, whereas clock buffers/generators are better suited for applications that need several individual clocks. Clock buffers can be used in conjunction with a XO reference to distribute multiple clocks at the same frequency and provide the lowest jitter implementation for a multi-output clock tree. Clock generators can synthesize multiple clocks at different frequencies, but sacrifice some jitter performance in comparison to XO+buffer clock trees.

Synchronous clocking is used in applications that require continuous communication and network-level synchronization, such as OTN (Optical Transport Networking), SONET/SDH, mobile fronthaul/backhaul, Synchronous Ethernet and HD SDI video transmission. These applications require transmitters and receivers to operate at the same frequency. By synchronizing all SerDes reference clocks to a highly accurate network reference clock (e.g. Stratum 3, GPS) synchronization across all nodes in the network is guaranteed. In these applications, low bandwidth PLL-based clocks provide wander and jitter filtering (jitter cleaning) to ensure network-level synchronization is maintained. In networking line card PLL applications, specialized jitter attenuating clocks or discrete VCO (Voltage-Controlled Oscillator)-based PLLs are the preferred clock solution for SerDes clocking. For optimal performance, a jitter attenuating clock should be placed at the end of the clock tree, directly driving the SerDes. Clock generators and buffers can be used to provide other system references.
Jitter

Jitter is a critical specification for timing components because excessive clock jitter can compromise system performance. There are three common types of clock jitter. Depending on the application, one type of jitter will be more important than others. Cycle-to-cycle jitter measures the maximum change in clock period between any two adjacent clock cycles, typically measured over 1,000 clock cycles. Period jitter is the maximum deviation in clock period with respect to an ideal period over a large number of cycles (10,000 clock cycles typ.). Both cycle-cycle jitter and period jitter are useful in calculating setup and hold timing margins in digital systems, so they are often figures of merit for CPU and SoC devices. Phase jitter is the figure of merit for high-speed SerDes applications. Phase jitter is a ratio of noise power to signal power calculated by integrating the clock single sideband phase noise across a range of frequencies offset from a carrier signal. Phase jitter is especially critical in FPGA and high-speed SerDes clocking applications in which excessive phase jitter can degrade the Bit-Error-Rate of the high-speed serial interface.

During clock tree design and component selection, it is important to evaluate devices based on maximum jitter performance. Typical jitter specifications do not guarantee device performance over all conditions, including process, voltage, temperature, and frequency variation. Maximum jitter provides a more comprehensive specification inclusive of these additional factors.

In addition, take special care to review jitter test conditions on timing device datasheets. Clock jitter performance varies across a wide range of conditions, including device configuration, operating frequency, signal format, input clock slew rate, power supply and power supply noise. Look for devices that fully specify jitter test conditions since they guarantee operation over a wider operating range.
Selection Criteria for Clocks and Oscillators

Once the basic clock tree architecture is determined, the next step is component selection. The table below summarizes the selection criteria that should be used for selecting clock and oscillator components for both free-running and synchronous clock trees. Look for features that simplify clock tree design to minimize BOM cost and complexity.

<table>
<thead>
<tr>
<th>Function</th>
<th>XO</th>
<th>VCXO</th>
<th>Clock Generator</th>
<th>Clock Buffer</th>
<th>Jitter Cleaning Clock/Network Synchronizer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Free-run operation</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Synchronous operation</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Clock Multiplication</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Clock Division</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Jitter Cleaning</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Design Complexity</td>
<td>Low</td>
<td>High</td>
<td>Medium</td>
<td>Low</td>
<td>Medium</td>
</tr>
<tr>
<td>Integration</td>
<td>Low</td>
<td>Low</td>
<td>High</td>
<td>High</td>
<td>High</td>
</tr>
</tbody>
</table>

Features That Simplify Clock Tree Design

- Small form factor
- Format translation
- Integrated input mux
- Glitchless switching between clocks at different frequencies
- Clock division
- Synchronous output clock disable
- Holdover
- Integrated power supply filtering

Table 1. Timing Component Selection Criteria

Estimating Clock Tree Jitter

Before a clock tree design is complete, the total clock tree jitter should be estimated to determine if there is sufficient system-level design margin. It is important to note that total clock tree RMS jitter is much less than the simple sum of data sheet jitter specifications from multiple components. The clock tree jitter can be defined by the following equation:

\[ T_j (RMS) = \sqrt{J_1^2 + J_2^2 + \cdots + J_n^2} \]  (Equation 1)

Where \( T_j \) = Total RMS jitter, \( J_n \) = individual device RMS jitter.

Note: This equation can be applied to calculating total period jitter and phase jitter, assuming the jitter distributions are Gaussian and uncorrelated. This equation should not be applied to cycle-cycle jitter because cycle-cycle jitter is expressed as a peak jitter number, not RMS.

Component jitter \( J_n \) can be estimated using datasheet jitter specifications or calculated from phase noise data. Silicon Labs offers an easy-to-use utility for converting clock phase noise to jitter. See [http://www.silabs.com/support/Pages/phase-noise-jitter-calculator.aspx](http://www.silabs.com/support/Pages/phase-noise-jitter-calculator.aspx) for more details. As mentioned above, be sure to use maximum jitter specifications to generate a conservative estimate of total clock tree jitter.
Simplifying Clock Trees

Many clock trees require specialty features in addition to basic clock generation and distribution. For example, the application may require format/level translation (e.g. 3.3V LVPECL to 2.5V LVDS), switching between two clocks at different frequencies, clock division, pin-selectable output enable control and CMOS drive strength (output impedance) control for EMI reduction. If designed discretely, implementing these functions adds significant cost and complexity to the clock tree design. Silicon Labs has developed a family of Si5330x universal buffers/translators that integrate format/level translation, clock muxing, clock division and other key clock tree building block functions. These devices replace multiple LVPECL, LVDS, CML, HCSL and LVCMOS buffers with a single clock buffer IC. In addition to simplified clock tree design, the Si5330x minimize BOM cost and complexity, simplify procurement and improve system performance.

![Traditional Clock Distribution](image1)

![Si5330x-Based Clock Distribution](image2)

Figure 2. Si5330x Simplifies Clock Tree Designs

Silicon Labs Timing Products

Silicon Labs offers the industry’s broadest portfolio of frequency flexible low jitter clock generators, clock buffers, jitter cleaning clocks, network synchronizers and XO/VCXOs. To learn more about Silicon Labs timing products, go to:

- Clocks: [https://www.silabs.com/products/timing/clocks](https://www.silabs.com/products/timing/clocks)