**Overview**

This white paper discusses **metastability**, a critical issue in FPGA designs, particularly when signals are transferred between **asynchronous clock domains**. It outlines the causes, implications, detection, and mitigation techniques for metastability, along with how to compute and improve the **Mean Time Between Failures (MTBF)**.

**Key Concepts**

**What is Metastability?**

* Occurs when a flip-flop's setup (tSU) or hold (tH) time is violated.
* Leads to undefined output that hovers between logic '0' and '1' (a metastable state).
* Happens primarily during **asynchronous signal transfer** across clock domains.

**When Does It Cause Failures?**

* If metastable output doesn’t resolve before the next logic stage samples it, inconsistent logic states may occur, causing system failures.

**Synchronization Registers**

* **Synchronizers** are chains of flip-flops that resample incoming asynchronous signals to reduce metastability risk.
* The **length** of the synchronizer affects the time available for metastable resolution.

**MTBF: Mean Time Between Failures**

MTBF Formula:

**C1, C2**: Constants based on device process and conditions.

**fCLK**: Clock frequency of the receiving domain.

**fDATA**: Toggle rate of incoming asynchronous signal.

**tMET**: Settling time (timing slack) available for metastability resolution.

**Higher MTBF = better reliability.** For instance, medical devices require much higher MTBF than consumer products.

**Improving MTBF**

**FPGA Architecture Enhancements**

* Faster processes improve MTBF, but lower voltages (e.g., in 65 nm or 40 nm nodes) may worsen it.

**Design Optimizations**

* Increase **tMET** by adding more flip-flops in synchronizers. e.g., 3-stage chains tMET(total) = tMET1 + tMET2 + tMET3
* **Worst-case synchronizer** dominates overall MTBF. Addressing the weakest link is crucial.

Finally, to improve the system’s MTBF, synchronizer’s length should be increased.

**Conclusion**

* Metastability is inevitable in asynchronous designs but can be **effectively mitigated** through proper synchronizer design and architecture-aware placement.
* Enhancing **tMET** or improving architecture constants (C2) significantly boosts system robustness.