Understanding Metastability in FPGAs

This white paper describes metastability in FPGAs, why it happens, and how it can cause design failures. It explains how metastability MTBF is calculated, and highlights how various device and design parameters affect the result.

Introduction

Metastability is a phenomenon that can cause system failure in digital devices, including FPGAs, when a signal is transferred between circuitry in unrelated or asynchronous clock domains. This paper describes metastability in FPGAs, explains why the phenomenon occurs, and discusses how it can cause design failures.

The calculated mean time between failures (MTBF) due to metastability indicates whether designers should take steps to reduce the chance of such failures. This paper explains how MTBF is calculated from various design and device parameters, and how both FPGA vendors and designers can increase the MTBF. System reliability can be improved by reducing the chance of metastability failures with design techniques and optimizations.

What Is Metastability?

All registers in digital devices such as FPGAs have defined signal timing requirements that allow each register to correctly capture data at its inputs and produce an output signal. To ensure reliable operation, the input to a register must be stable for a minimum time before the clock edge (register setup time or t_{SU}) and for a minimum time after the clock edge (register hold time or t_H). The register output then is available after a specified clock-to-output delay (t_{CO}). If a data signal transition violates a register's t_{SU} or t_H requirements, the output of the register may go into a metastable state. In a metastable state, the register output hovers at a value between the high and low states for some period of time, which means the output transition to a defined high or low state is delayed beyond the specified t_{CO}.

In synchronous systems, the input signals must always meet the register timing requirements, so metastability does not occur. Metastability problems commonly occur when a signal is transferred between circuitry in unrelated or asynchronous clock domains. The designer cannot guarantee that the signal will meet t_{SU} and t_H requirements in this case, because the signal can arrive at any time relative to the destination clock. However, not every signal transition that violates a register's t_{SU} or t_H results in a metastable output. The likelihood that a register enters a metastable state and the time required to return to a stable state vary depending on the process technology used to manufacture the device and on the operating conditions. In most cases, registers will quickly return to a stable defined state.

A register sampling a data signal at a clock edge can be visualized as a ball being dropped onto a hill, as shown in Figure 1. The sides of the hill represent stable states—the signal's old and new data values after a signal transition—and the top of the hill represents a metastable state. If the ball is dropped at the top of the hill, it might balance there indefinitely, but in practice it falls slightly to one side of the top and rolls down the hill. The further the ball lands from the top of the hill, the faster it reaches a stable state at the bottom.

If a data signal transitions after the clock edge and the minimum t_H, it is analogous to the ball being dropped on the "old data value" side of the hill, and the output signal remains at the original value for that clock transition. When a register's data input transitions before the clock edge and minimum t_{SU}, and is held beyond the minimum t_H, it is analogous to the ball being dropped on the "new data value" side of the hill, and the output reaches the stable new state quickly enough to meet the defined t_{CO} time. However, when a register's data input violates the t_{SU} or t_H, it is analogous to the ball being dropped on the hill. If the ball lands near the top of the hill, the ball takes too long to reach the bottom, which increases the delay from the clock transition to a stable output beyond the defined t_{CO}.

Figure 1. Metastability Illustrated as a Ball Dropped on a Hill

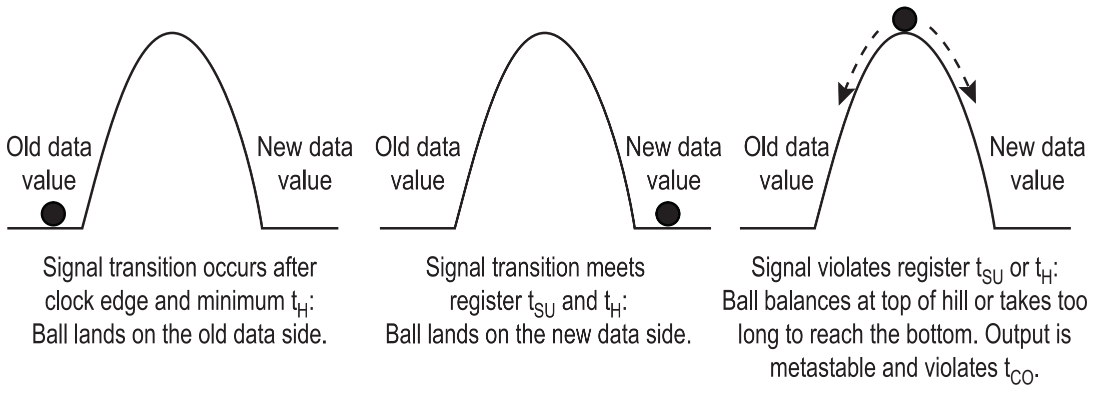
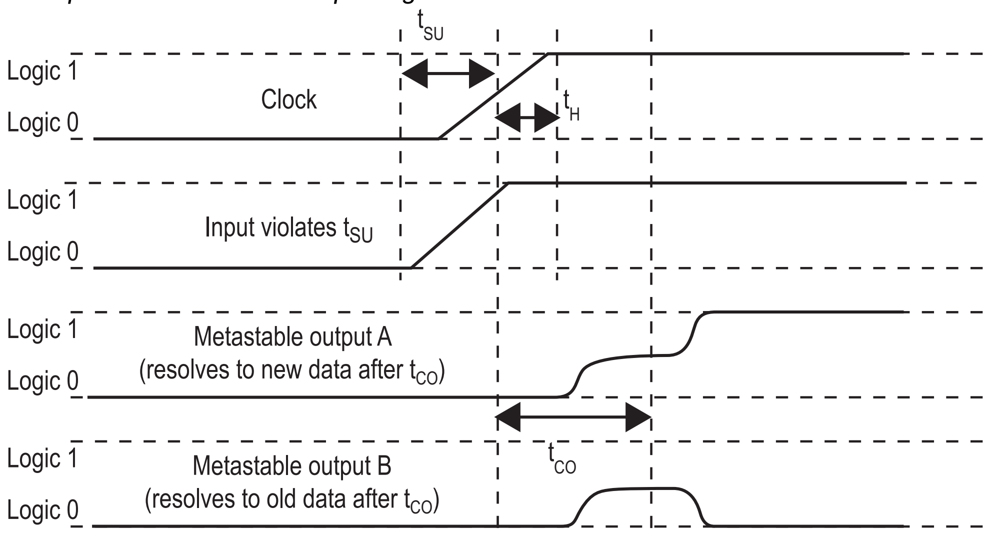

Old data value
New data value
Signal transition occurs after clock edge and minimum t_H: Ball lands on the old data side.
Old data value
New data value
Signal transition meets register t_{SU} and t_H: Ball lands on the new data side.
Old data value
New data value
Signal violates register t_{SU} or t_H: Ball balances at top of hill or takes too long to reach the bottom. Output is metastable and violates t_{CO}.

Figure 2 illustrates metastable signals. The input signal transitions from a low state to a high state while the clock signal transitions, violating a register's t_{SU} requirement. The data output signal examples start in the low state and go metastable, hovering between the high and low states. The signal output *A* resolves to the input data's new logic 1 state, and output *B* returns to the data input's original logic 0 state. In both cases, the output transition to a defined 1 or 0 state is delayed beyond the register's specified t_{CO}.

Figure 2. Examples of Metastable Output Signals


t_{SU}
Logic 1
Clock
t_H
Logic 0
Logic 1
Input violates t_{SU}
Logic 0
Logic 1
Metastable output A (resolves to new data after t_{CO})
Logic 0
t_{CO}
Logic 1
Metastable output B (resolves to old data after t_{CO})
Logic 0

When Does Metastability Cause Design Failures?

If the data output signal resolves to a valid state before the next register captures the data, then the metastable signal does not negatively impact the system operation. But if the metastable signal does not resolve to a low or high state before it reaches the next design register, it can cause the system to fail. Continuing the ball and hill analogy, failure can occur when the time it takes for the ball to reach the bottom of the hill (a stable logic value 0 or 1) exceeds the allotted time, which is the register's t_{CO} plus any timing slack in the path from the register. When a metastable signal does not resolve in the allotted time, a logic failure can result if the destination logic observes inconsistent logic states, that is, different destination registers capture different values for the metastable signal.

Synchronization Registers

When a signal transfers between circuitry in unrelated or asynchronous clock domains, it is necessary to synchronize this signal to the new clock domain before it can be used. The first register in the new clock domain acts as a synchronization register.

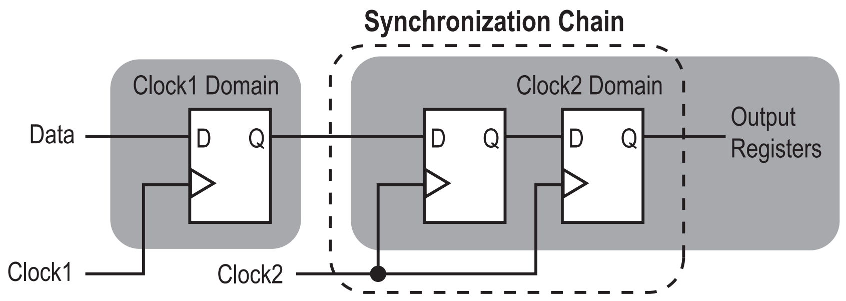
To minimize the failures due to metastability in asynchronous signal transfers, circuit designers typically use a sequence of registers (a synchronization register chain or synchronizer) in the destination clock domain to resynchronize the signal to the new clock domain. These registers allow additional time for a potentially metastable signal to resolve to a known value before the signal is used in the rest of the design. The timing slack available in the synchronizer register-to-register paths is the time available for a metastable signal to settle, and is known as the available metastability settling time.

A synchronization register chain, or synchronizer, is defined as a sequence of registers that meets the following requirements:

- The registers in the chain are all clocked by the same or phase-related clocks
- The first register in the chain is driven from an unrelated clock domain, or asynchronously
- Each register fans out to only one register, except the last register in the chain

The length of the synchronization register chain is the number of registers in the synchronizing clock domain that meet the above requirements. Figure 3 shows a sample synchronization chain of length two, assuming the output signal feeds more than one register destination.

Figure 3. Sample Synchronization Register Chain



Note that any asynchronous input signals, or signals that transfer between unrelated clock domains, can transition at any point relative to the clock edge of the capturing register. Therefore the designer cannot predict the sequence of a signal's transitions or the number of destination clock edges until the data transitions. For example, if a bus of asynchronous signals is transferred between clock domains and synchronized, the data signals could transition on different clock edges. As a result, the received values of the bus data could be incorrect.

The designer must accommodate this behavior with circuitry such as dual-clock FIFO (DCFIFO) logic to store the signal values, or hand-shaking logic. FIFO logic uses synchronizers to transmit control signals between the two clock domains, then data is written and read with dual-port memory. Altera offers a DCFIFO megafunction for this operation, which includes various levels of latency and metastability protection for the control signals. Otherwise, if an asynchronous signal acts as part of hand-shaking logic between two clock domains, control signals indicate when data can be transferred between clock domains. In this case, synchronization registers are used to ensure that metastability will not interfere with the reception of control signals and that the data has enough settling time for any metastable conditions to resolve before the data is used. In a properly-designed system, the design functions correctly as long as each signal resolves to a stable value before it is used.

Calculating Metastability MTBF

The mean time between failures, or MTBF, due to metastability provides an estimate of the average time between instances when metastability could cause a design failure. A higher MTBF (such as hundreds or thousands of years between metastability failures) indicates a more robust design. The required MTBF depends on the system application. For example, a life-critical medical device requires a higher MTBF than a consumer video-display device. Increasing the metastability MTBF reduces the chance that signal transfers will cause any metastability problems on the device.

The metastability MTBF for a specific signal transfer, or all the transfers in a design, can be calculated using information about the design and the device characteristics. The MTBF of a synchronizer chain is calculated with the following formula and parameters:

$$MTBF = \frac{e^{t_{MET}/C_2}}{C_1 \cdot f_{CLK} \cdot f_{DATA}}$$

The C_1 and C_2 constants depend on the device process and operating conditions.

The f_{CLK} and f_{DATA} parameters depend on the design specifications: f_{CLK} is the clock frequency of the clock domain receiving the asynchronous signal and f_{DATA} is the toggling frequency of the asynchronous input data signal. Faster clock frequencies and faster-toggling data reduce (or worsen) the MTBF.

The t_{MET} parameter is the available metastability settling time, or the timing slack available beyond the register's t_{CO}, for a potentially metastable signal to resolve to a known value. The t_{MET} for a synchronization chain is the sum of the output timing slacks for each register in the chain.

The overall design MTBF can be determined by the MTBF of each synchronizer chain in the design. The failure rate for a synchronizer is 1/MTBF, and the failure rate for the entire design is calculated by adding the failure rates for each synchronizer chain, as follows:

$$failure\_rate_{design} = \frac{1}{MTBF_{design}} = \sum_{i=1}^{number\ of\ chains} \frac{1}{MTBF_i}$$

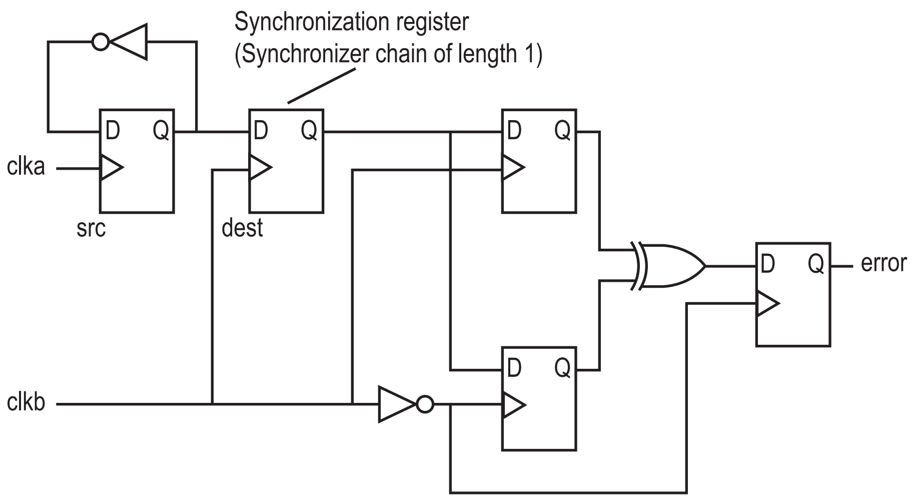
The design metastability MTBF is then 1/$failure\_rate_{design}$.

Designers using Altera® FPGAs do not have to perform these calculations manually because Altera's Quartus® II software incorporates the metastability parameters within the tool. The Quartus II software reports the MTBF for identified synchronization chains as well as providing an overall design metastability MTBF.

Characterizing Metastability Constants

FPGA vendors can determine the constant parameters in the MTBF equation by characterizing the FPGA for metastability. The difficulty with this characterization is that MTBFs for typical FPGA designs are in years, so measuring the time between metastability events using real designs under real operating conditions is impractical. To characterize the device-specific metastability constants, Altera uses a test circuit designed to have a short, measurable MTBF, as shown in Figure 4.

Figure 4. Test Circuit Structure for Metastability Characterization



In this design, clka and clkb are two unrelated clock signals. The data input to the synchronizer toggles every clock cycle (a high f_{DATA}). The synchronizer has length 1, because the single synchronizing register feeds two destination registers. The destination registers capture the output of the synchronizer one clock cycle later and one half-clock cycle later. If the signal goes metastable before resolving at the next clock edge, the circuit detects that the sampled signals are different, and outputs an error signal. This circuitry detects a high proportion of the metastability events that occur at the half-clock cycle time.

This circuit is replicated throughout the device to reduce the effect of any local variation, and each instance is tested consecutively to eliminate any noise coupling. Altera measures each test structure for one minute and records the error count. The test is performed at different clock frequencies, and the MTBF versus t_{MET} results are plotted on a logarithmic scale. The C_2 constant corresponds to the slope of the trend line for the experimental results, and the C_1 constant scales the line linearly.

Improving Metastability MTBF

Due to the exponential e^{t_{MET}/C_2} factor in the MTBF equation, the t_{MET}/C_2 term has the largest effect on the MTBF calculation. Therefore metastability can be improved by optimizing the device's C_2 constant with architecture enhancements, or optimizing the design to increase the t_{MET} in the synchronization registers.

FPGA Architecture Enhancements

The metastability time constant C_2 in the MTBF equation depends on various factors related to the process technology used to manufacture the device, including the transistor speed and the supply voltage. Faster process technologies and faster transistors allow metastable signals to resolve more quickly. As FPGAs have migrated from 180-nm process geometries to 90 nm, the increase in transistor speed usually improves metastability MTBF. Therefore, metastability has not been a major concern for FPGA designers.

However, as the supply voltage reduces with reduced process geometries, the threshold voltage for the circuit does not decrease proportionally. When a register goes metastable, its voltage is approximately one-half of the supply voltage. With a reduced power supply voltage, the metastable voltage level is closer to the threshold voltage in the circuit. When these voltages get closer together, the gain of the circuit is reduced and the registers take longer to transition out of metastability. As FPGAs enter the 65-nm process geometry and lower, with power supplies at 0.9V and lower, the threshold voltage consideration is becoming more important than the increase in transistor speed. Therefore, metastability MTBFs generally get worse unless the vendor designs the FPGA circuitry to improve metastability robustness.

Altera uses metastability analysis of the FPGA architecture to optimize the circuitry for improved metastability MTBF. Architecture improvements in Altera's 40-nm Stratix® IV FPGA architectures and new device development have improved the metastability robustness results by reducing the MTBF C_2 constant.

Design Optimizations

The exponential factor in the MTBF equation means that an increase in the design-dependent t_{MET} value increases a synchronizer MTBF exponentially. For example, if the C_2 constant for a given device and set of operating conditions is 50 ps, then an increase of just 200 ps in the t_{MET} makes the exponent 200/50 and increases the MTBF by factor e^4, or more than 50 times, while an increase of 400 ps multiplies the MTBF by e^8, or almost 3000 times.

In addition, the chain with the worst MTBF has a major affect on the design MTBF. For example, consider two different designs that have ten synchronizer chains. One design has ten chains with the same MTBF of 10,000 years, and the other has nine chains with MTBF of a million years but one chain with MTBF of 100 years. The failure rate for the design is the sum of the failure rates for each chain, where the failure rate is 1/MTBF. The first design has a metastability failure rate of 10 chains × 1/10,000 years = 0.001, therefore the design MTBF is 1000 years. The second design has a failure rate of 9 chains × 1/1,000,000 + 1/100 = 0.01009 and the design MTBF is about 99 years—just slightly less than the MTBF of the worst chain.

Put another way, one badly designed or implemented synchronization chain dominates the design's overall metastability MTBF. Because of this effect, it is important to perform metastability analysis for all asynchronous signals and clock domain transfers. The designer or tool vendor can have a very significant impact on a design MTBF by improving the t_{MET} for the synchronizer chains with the worst MTBF.

To improve metastability MTBF, designers can increase t_{MET} by adding extra register stages to synchronization register chains. The timing slack on each additional register-to-register connection is added to the t_{MET} value. Designers commonly use two registers to synchronize a signal, but Altera recommends using a standard of three registers for better metastability protection. However, adding a register adds an additional latency stage to the synchronization logic, so designers must evaluate whether that is acceptable.

If a design uses the Altera FIFO megafunction with separate read and write clocks to cross-clock domains, designers can increase the metastability protection (and latency) for better MTBF. Altera's Quartus II MegaWizard™ Plug-In Manager offers an option to choose increased metastability protection option with three or more synchronization stages.

Quartus II software also offers industry-leading metastability analysis and optimization features to increase the t_{MET} on synchronization register chains. When synchronizers are identified, the software places synchronization registers closer together to increase the output timing slack available in the synchronizer chain, and then reports the metastability MTBF.

For more information about analyzing and improving MTBF in Altera FPGAs, refer to the "Managing Metastability with the Quartus II Software" chapter in volume 1 of the *Quartus II Handbook*.

Conclusion

Metastability can occur when signals are transferred between circuitry in unrelated or asynchronous clock domains. The mean time between metastability failures is related to the device process technology, design specifications, and timing slack in the synchronization logic. FPGA designers can improve system reliability and increase metastability MTBF by increasing the t_{MET} with design techniques that add timing slack in synchronization registers. Altera characterizes the MTBF parameters for its FPGAs and improves metastability MTBF with device technology improvements. Designers using Altera FPGAs can take advantage of Quartus II software features to report metastability MTBF for their design, and optimize design placement to increase MTBF.

Further Information

- *Managing Metastability with the Quartus II Software:* www.altera.com/literature/hb/qts/qts_qii51018.pdf
- *AN 473: Using DCFIFO for Data Transfer Between Asynchronous Clock Domains:* www.altera.com/literature/an/an473.pdf

Acknowledgments

- Jennifer Stephenson, Applications Engineer, Member of Technical Staff, Software Applications Engineering, Altera Corporation
- Doris Chen, Advanced Software Engineer, Software and Systems Engineering, Altera Corporation
- Ryan Fung, Senior Member of Technical Staff, Software and Systems Engineering, Altera Corporation
- Jeffrey Chromczak, Senior Software Engineer, Software and Systems Engineering, Altera Corporation

101 Innovation Drive
San Jose, CA 95134
www.altera.com