Metastability in logic circuits

Many faults in digital circuit design, specially those using flip flops can be traced to the phenomenon of ***metastability***. It is something that for a long time was not even recognized as an error source. However, a number of design engineers including the author of this article have personally experienced the tragedy of logic failure owing to metastability. In most instances the failure is hard to spot and can take its toll on the engineer trying to debug a malfunctioning logic circuit. Sometimes the problem is made even more catastrophic because the circuit may function quite well under certain conditions and fail under others. It can literally drive the design engineer up the wall! In analog and mixed signal design, whether it is a lower frequency circuit or a higher frequency circuit the causes and effects of metastability remain about the same. For those readers who may have experienced this and wish to investigate further, please read on.

In order to elaborate on the concept of metastability please refer to Figure 1.0 below.

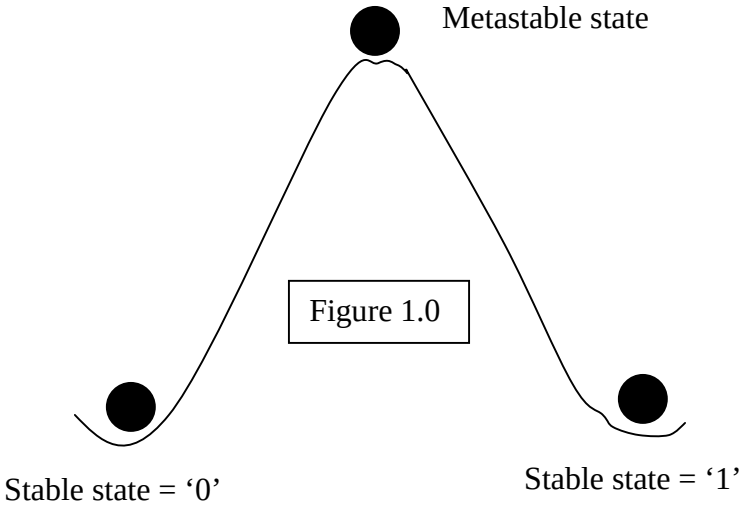

Figure 1.0

The analogy is to a hill. Lets assume that this metaphor is for a D type Flip Flop. When the FF is working properly it will exhibit two stable states. Usually referred to as a '0' or a '1' state.

However certain operating regimes can cause the temporary establishment of a third state called a metastable state. This state is shown by the ball at the peak of the hill. The ball can obviously roll either to the '0' side or the '1' side. This type of behavior can at best cause long delays in the logic or at worst cause the logic to become completely useless. Therefore an understanding of metastability can only help the designer to: (1) help in debugging an errant logic circuit or, (2) design for robustness.

Some of the factors that can cause metastability are listed and described below.

Generally, whenever setup and hold violation time occurs, metastability occurs. However, this is not always the case. Other factors can also cause it. Metastability is usually common when the logic block signals violate setup and hold time requirements, Under this assumption the following are a majority of reasons for metastability.

A. When the input signal is asynchronous.
B. When the clock skew or slew is too much (rise and fall times out of tolerance).
C. When interfacing two domains operating at different frequencies or at the same frequency but with different phase.
D. When the nominal circuit delay is such that FF data input changes in the critical setup and hold time window.

Completely eliminating metastability is not possible, so designers need to be able to limit the possibility of metastable behavior to a significant degree. Some of the ways to do this include *using only one clock*, using *faster flipflops*, *decrease* the asynchronous input frequency, and *use synchronization hardware*

.

When more than one clock is used, the time window in which that the input is vulnerable to metastability occurs more often. Using faster flipflops decreases the setup and hold times of the flipflop, which in turn decreases the time window that the flipflop is vulnerable to metastability. When the input frequency is decreased, the chances of the input changing during the setup and hold time also decreases. Figure 2 below represents a type of synchronous hardware that can also reduce metastability.

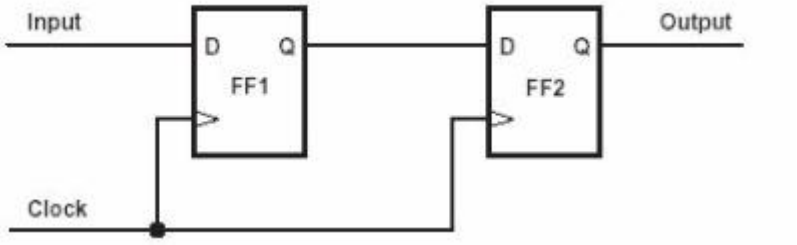

Figure 2.0 Synchronizer.

In reality, one cannot avoid metastability and increased clock-to-Q delays in synchronizing asynchronous inputs, without the use of tricky self-timed circuits.

In the simplest case, designers can learn to tolerate metastability by making sure the clock period is long enough to allow for the resolution of quasi-stable states and for the delay of whatever logic may be in the path to the next flip-flop. This approach, while simple, is rarely practical given the performance requirements of most modern designs.

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.

An expression that can be used to calculate MTBF is shown below.

$$MTBF = \exp(T \cdot TX)/\{ fclk \cdot fin \cdot T0)$$

Here,

T , T0 = fitting parameters dependent on
Process and design type. These are derived from actual metastabilty measurements.

TX = the time delay between the clock edge and a stable response on the output when metastability occurs .

Examples:
Standard TTL, T =0.74 , T0 = 2.9E-4
LS T =0.74, T0 = 4.8e-3
The units for T and T0 are 1/ns.

fclk = clock frequency
fin = frequency of the asynchronous input signal.

Cascaded systems generate multiplicative MTBFs. These expressions and other like it provide some numbers to play with. However, engineering judgment must come into play

as well. A priori the recommendation should be to derate the MTBF by at least 2X *until proven different by measurement.*