EE552

Advanced Logic Design
and
Switching Theory

Metastability

by

Ashirwad Bahukhandi

(Ashirwad Bahukhandi)

bahukhan@usc.edu

This is an overview of what metastability is, ways of interpreting it, the issues concerning it and the remedies suggested. I have mentioned all the references that I have gone through.

Also note that, keeping in mind the purview of EE552, in this report, I have tried to provide the logic design perspective of metastability rather than the circuit level design viewpoint. This could be a potential additional reading assignment to the metastability material provided in the class. The class material is not covered here to avoid repetition.

Moreover, I would like to mention that I have compiled this report in personal capacity and have not taken any assistance from anyone. I give full permission for this report to be posted on the class web-site.

Ashirwad Bahukhandi

INTRODUCTION

As system designers continue to push the upper bound of performance, understanding the metastability operation of flip-flops is important to reliability. Good synchronous design practice or careful evaluation of device characteristics can achieve high reliability. As the speed of designs increases to 50 MHz and above, it's quite understandable that the design is hard pressed to meet the timing requirements. All sequential components in a design have to satisfy certain minimal timing requirement for data arrival to ensure a good logic 1 or a good logic 0. The data should arrive a minimum time before the active edge of the clock (and remain stable) for the clock to latch a valid logic of the data (setup time) and similarly this data should also remain stable for a minimum specified time after the active edge of the clock (hold time). These specs vary according to logic device. Any violation of these timing requirements may lead to erroneous data being latched. So it's not surprising that our everyday simple digital designs can be marred with timing errors. These errors are so subtle that we more often than not tend to ignore them. However, they make the nets in our design candidates of a potential disaster. The frequency of these timing mis-matches is so small; we call them "glitches" and generally dismiss them out of hand. Since at the occurrence of these violations, the data in the flip-flop is unknown, this condition is called "metastability", and is clearly a profound issue and good designers pay due importance to this.

Once the flip-flop has entered the metastable state, the probability that it will still be metastable some time later has been shown to be an exponentially decreasing function. Because of this property, a designer can simply wait for some added time after the specified propagation delay before sampling the flip-flop output so that he can be assured that the likelihood of metastable failure is remote enough to be tolerable. On the other hand, one consequence of this is that there is some probability (albeit vanishingly small) that the device will remain in a metastable state forever. The designer needs to know the characteristics of metastability so that he can determine how long he must wait to achieve his design goals.

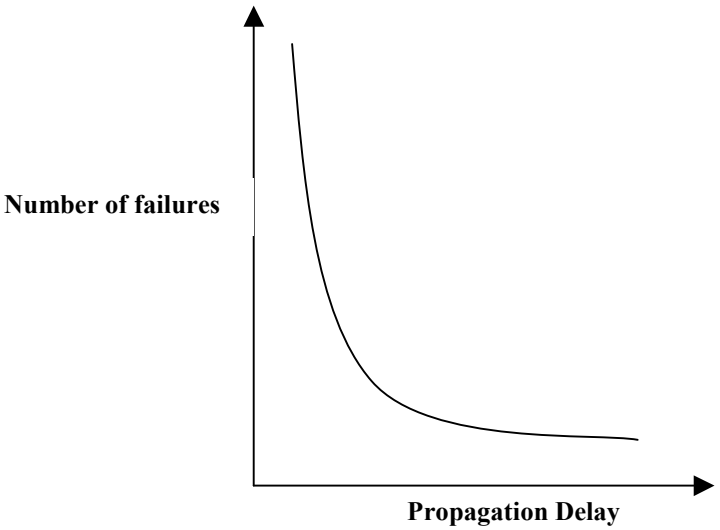

Fig.1 Number of failures v/s Propagation delay

WHAT IS METASTABILITY?

Metastability in digital systems occurs when two asynchronous signals combine in such a way that their resulting output goes to an indeterminate state. A common example is the case of data violating the setup and hold specifications of a latch or a flip-flop. In an ideal world, where all logic designs are synchronous and all inputs are tied to the system clock, metastability would not be a concern because all timing conditions for the flip-flops would be met. However, in most of the design, the data is asynchronous w.r.t. the clock making the flop a potential candidate for metastability as there's no reasonable way to insure that the changing asynchronous data will meet the flop's setup time. Occasionally – not often - the latched data will be corrupt. So the designer has to take care of these violations.

The duration of the metastable condition is a probabilistic phenomenon, and therefore there is no guaranteed maximum time. One can't build a bistable device such as a flip-flop that cannot go metastable.
Metastability can appear as a flip-flop that switches late or doesn't switch at all. It can present a brief pulse at a flip-flop output (called a runt pulse) or cause flip-flop output oscillations. Any of these conditions can cause system failures.

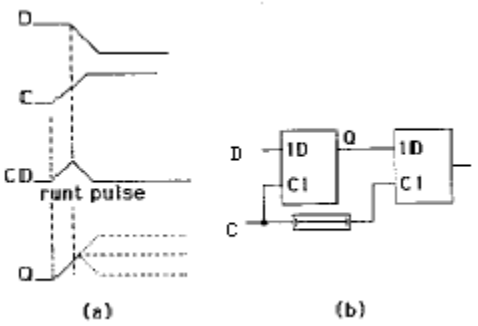

Fig. 2 Phenomenon of metastability: (a) runt pulse production; (b) a circuit to reduce the effect of metastability.

[13]
For a simple CMOS latch, valid data must be present on the input for a specified period of time before the clock signal arrives (setup time) and must remain valid for a specified period of time after the clock transition (hold time) to assure that the output functions predictably. This leaves a small window of time with respect to the clock (t 0) during which the data is not allowed to change. If a data edge occurs within this aperture, the output may go to an intermediate level and remain there for an indefinite amount of time before resolving itself either high or low, as illustrated in Figure 3.

This metastable event can cause a failure only if the output has not resolved itself by the time that it must be valid for use (for example, as an input to another stage); therefore, the amount of resolve time allowed a device plays a large role in calculating its failure rate. Whenever there is any such violation, the output voltage is anywhere between a logic high and a logic zero. In such a condition the flip-flop takes additional time to settle to a stable output. And this stable output depends upon the process technology, manufacture

and environment conditions that may force the output to go to a particular value. There is no way that the final state can be predicted.

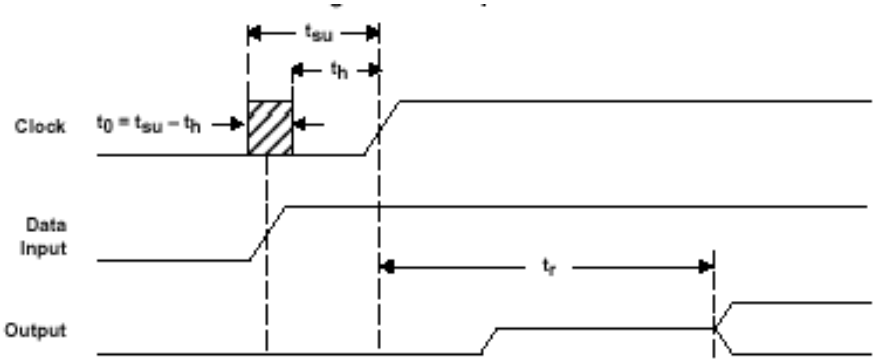

Fig 3 [4]

This operation is analogous to a ball rolling over a hill (Fig 4). Each side of the hill represents a stable state, and the top of the hill represents the metastable state. Whenever the flip-flop satisfies the setup and hold timings, the output achieves a stable state (either 1 or 0). However, when ever there is any violation, the correct data might not be latched (either at the slave part or the master part of the f/f) as the f/f is marginally triggered and the time required for the proper propagation of the data through the f/f is not met and the output is metastable. This is analogous to the ball at the top of the mountain in theory; a flip-flop in this quasi-stable hilltop state could remain there indefinitely – but in reality it won't. Just as the slightest air current would eventually cause a ball on the illustrated hill to roll down one side or the other, thermal and induced noise will jostle the state of the flip-flop causing it to move from the quasi-stable state into either the logic 0 or logic 1 state. Hence, the output is random. In any case the CP-Q delay of the f/f is increased. The extra delay may be ten or twenty times longer than the normal clk-Q delay. This extra time is called the metastable resolution time. However, metastability may not always result in unpredictable output. If provided sufficient time with proper excitation, the f/f can infact settle to a stable state. But in the post GHz designs, time is at a premium, so the occurrence of the metastability itself has to be taken care of.

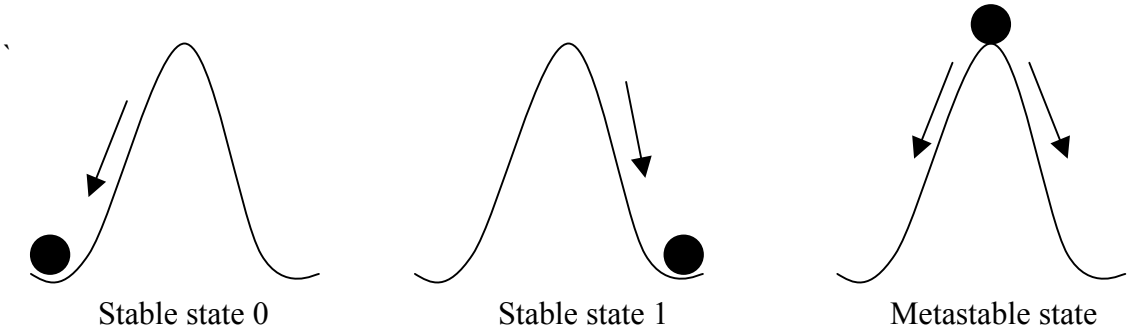

Fig. 4

In other words, in a device, two stable equilibrium states are potential-energy minimums. Between the two minimums is a potential-energy maximum (analogous to a hill).

Because the slope of the energy curve is 0 at the maximum, the maximum is also an equilibrium state, although an unstable one.

Metastable events are associated with data transitions occurring close to the active edge of the clock. Figure 3 shows an example of data changing within a timing window that will result in a timing violation. Because a timing violation has occurred, the flip-flop will exhibit erratic behavior. The erratic behavior manifests itself in the form of an extended propagation delay with an unpredictable resolution of the Q output (of f/f). This metastable event can cause a failure only if the output has not resolved itself by the time that it must be valid for use (e.g. as an input the another stage); therefore, the amount of resolve time allowed for a device plays a large role in calculating its failure rate.

WHAT ARE THE CASES, WHEN METASTABILITY OCCURS?

As we have seen that whenever setup and hold violation time occurs, metastability occurs, so it is to be seen when does this signal violate this timing requirement. [9]

- When the input signal is a asynchronous signal
- When the clock skew is more (rise time and fall time is more then the tolerable values).
- When interfacing two domains operating at two different frequency.
- When the combinational delay is such way that, it changes flip-flop's input in the required window (setup + hold window)

HOW IS METASTABILITY DESCRIBED?

In order to define the metastability characteristics of a device, three things must be known: first, what is the likelihood that the device will enter a metastable state? This propensity is defined by the parameter "To ". Second, once the device is in a metastable state, how long would it be expected to remain in that state? This parameter is T and is simply the exponential time constant of the decay rate of the metastability. It is sometimes called the metastability time constant. The final parameter is the measured propagation delay of the device.

Because metastability formulas aren't standardized, one has to read application notes carefully to understand the manufacturer's definition of each parameter. Metastability is typically described by four measurements of flip-flop performance — MTBF, T, To and t_r. MTBF is the "mean-time-between-failure" of a flip-flop.

$$\mathrm{MTBF} = \frac{\exp(t_r / \mathrm{T})}{T_0 \cdot f_{tn} \cdot f_{clock}}$$

where t_r is metastability resolution time, maximum time the output can remain metastable without causing synchronizer failure. T and T_0 are constants that depend on the electrical characteristics, process technology and the internal design of the flip-flop, f_{tn} is the frequency of the asynchronous input and f_{clock} is the frequency of the sampling clock.

The exponential term in the equation describes the probability that a metastable condition will last for time t'. As one increases time t' that one waits before looking at a f/f's output, the likelihood of seeing unresolved metastability is exponentially decreased. As seen from the expression, MTBF is MTBF is inversely proportional to the clock rate (f_{clock}) and the data rate (f_{tn}). In designs having asynchronous data, most designers do not know their data rate, so it is difficult to estimate the MTBF accurately. Usually, a small time period is considered (10 seconds, for example) and the number of clocks and data transitions during the small time is used to define f_{clock} and f_{tn}. As the time delay is increased, the number of failures decreases dramatically. By counting the number of failures over time, MTBF can be directly calculated. The values are derived by a formula that includes counts of the number of failures and the time delays for sampling.

These are determined by plotting the natural log of MTBF versus t_{res} and performing a linear regression analysis on the data. The y-intercept and slope of the resulting line determines the values of To and T. Their formulae are as follows:

$$1/T = \frac{\Delta \ln(\mathrm{MTBF})}{\Delta t_{res}}$$

$$To = \frac{e^{(tres/T)}}{\mathrm{MTBF} \times f_{clock} \times f_{tn}}$$

This method of computing MTFB and in-fact metastabilty has long been used in the industry.

HOW TO MINIMIZE METASTABILITY?

In the simplest case, designers can tolerate metastability by making sure the clock period is long enough to allow for the resolution of quasi-stable states as well as 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 following few methods are used to avoid metastability:

1. Synchronize any asynchronous input through one path that has at least one and preferably two flip-flops in series. The flip-flops should be running on the same edge of your system clock as the rest of the circuit. This will limit the area of potential problems to one path instead of several, and minimize the possibility of metastability entering the main part of the circuit. Use buffered flip-flops, or un-buffered flip-flops with minimum load. The second flop's output will be correct after two clocks, since the odds of two metastable events occurring back-to-back is almost nil. In a practical circuit, cascading two flip-flops practically squares the probability of failure. With two flip-flops, and at reasonable data rates, errors occur millions or even billions of years apart. Good enough for most systems. But "correct" means the second stage's output will not be metastable: it's not oscillating, nor is it at an illegal voltage level.

There's still an equal chance the value will be in either legal logic state. Thus, this is a very powerful technique. [8]

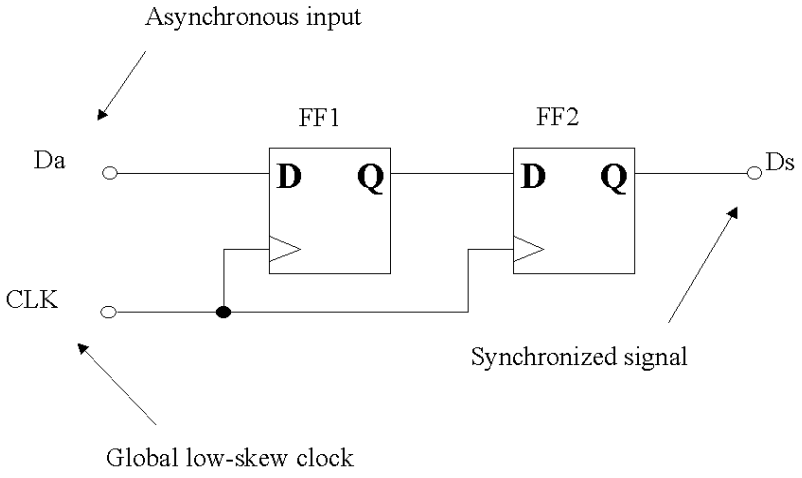

Fig. 5 Synchronizer

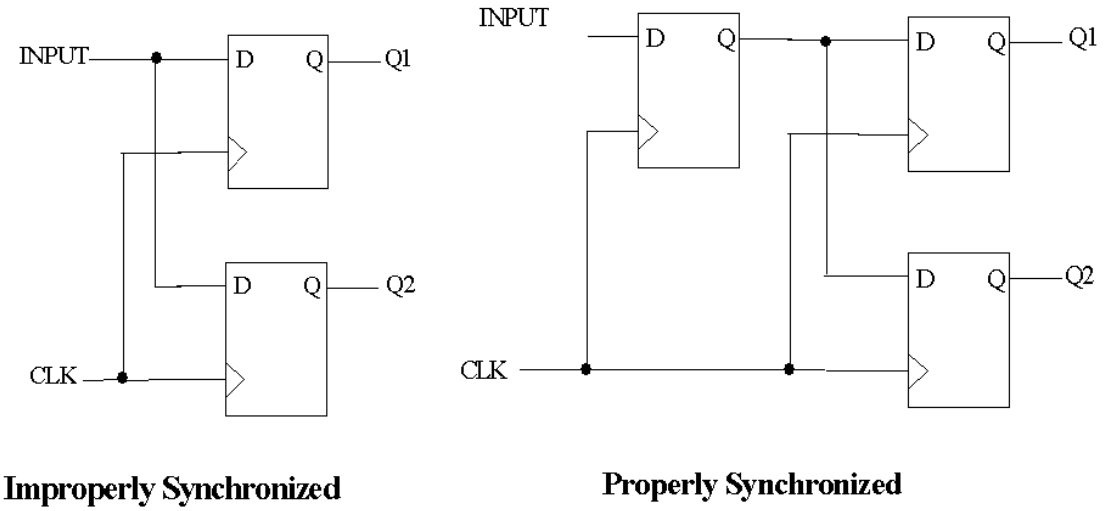

Fig. 6 Synchronizing an asynchronizing input

2. Design any state machines whose operation is affected by these “synchronized” signals to follow a gray code pattern between states controlled by these signals. Gray

Code is a counting scheme where only a single bit changes between numbers, as follows:

000
001
011
010
110
111
101
100

Gray code makes sense if, and only if, your code reads the device faster than it is likely to change, and if the changes happen in a fairly predictable fashion-like counting up. Then there's no real chance of more than a single bit changing between reads; if the inputs go metastable, only one bit will be wrong. The result will still be reasonable. This will prevent the state machine from "taking off" to unwanted states should the synchronizing flip-flops be metastable. [3]

3. As mentioned earlier, ensure that setup time of the destination flip-flop is met. This will avoid the creation of metastable conditions inside the circuit and minimize the propagation of any should they occur.

4. Compute a parity or checksum of the input data before the capture register. Latch that into the register as well. Have the code compute parity and compare it to that read. If there's an error, do another read. [3]

5. Use metastability hardened Flip-flops (Their explanation is beyond the scope of this report). [11]

Some designs will never have a metastability problem. It always stems from violating set-up or hold times, which in turn comes from either poor design or asynchronous inputs. All of this discussion has revolved around asynchronous inputs, when the clock and data are unrelated in time. Be wary of anything not slaved to the clock e.g. interrupts in CPUs are not synchronous to processor clock. Be sure that these signal itself and the vector-generating logic, don't violate the processor's set-up and hold times.

Bad design, though, can plague any electronic system. Every logic component takes time to propagate data. When a signal traverses many devices, the delays can add up significantly. If the data then goes to a latch, it's quite possible that the delays may cause the input to transition at the same time as the clock. Instant metastability. Designers are generally pretty careful to avoid these situations, though.

By following these few precautions, the circuit will be resistant to the effects of metastability and more reliable.

EVEN AFTER ALL THIS METASTABILITY MIGHT BE REQUIRED!!!

There are instances when metastability might be required. A few cases are enlisted as follows: [2]

- Laboratory investigation of the metastable phenomenon in flip-flops
- Demonstration of quantum principles in introductory physics lab
- Development of solid-state random number generators
- Use of random number generator circuitry to produce multi-latches with mutually entangled metastable states from which one could build solid- state quantum ALUs

CONCLUSION

Metastability is unavoidable in asynchronous systems but careful attention to design can usually prevent the problem of violating setup and hold times. The metastability characteristics of a device depend upon the process technology used for its design and the environmental conditions. They have become increasingly prevalent at higher operating frequencies. When higher frequencies are used, extreme care must be taken to ensure that system reliability is not adversely affected due to inadequate synchronization methods. Various semiconductor companies employ different methods to tackle the problems arising due to metastability, few of which have been discussed. No matter what method is used, these failures must be accounted for in the design of asynchronous digital circuits.

References:

1. **http://www.xilinx.com/xcell/xl22/xl22_30.pdf**
2. **http://www.signalintegrity.com/news/4_4.htm**
3. **http://www.embedded.com/story/OEG20010718S0067**
4. **http://www-s.ti.com/sc/psheets/scza004a/scza004a.pdf**
5. **http://archives.e-insite.net/archives/ednmag/reg/1994/062394/13df2.htm**
6. **http://www.caa.lcs.mit.edu/~devadas/6.004/Lectures/lect7/sld014.htm**
7. **http://rk.gsfc.nasa.gov/richcontent/General_Application_Notes/mestablestates/xilinx_metastable_considerations.pdf**
8. **http://www.eetkorea.com/ARTICLES/2001APR/2001APR23_PL_CT_AN.PDF**
9. **http://www.deeps.org/Tidbits/meta.html**
10. **http://www.philipslogic.com/support/appnotes/an219.pdf**
11. **http://mikro.ee.tu-berlin.de/ifm/AW/HandOuts/ehlert.pdf**
12. **http://www.chipcenter.com/eexpert/hjohnson/hjohnson006.html**
13. **Class handout on metastability.**
14. **http://www.altera.com/literature/an/an042.pdf**