Metastability

What is it?
- "meta" means "between"

-The unpredictable behavior of a flip-flop or latch where its output assumes values between logic 0 and 1 for an unusually long period or even oscillates.

-In CMOS circuits it is often observed as an unusually long clock to Q delay.

What causes it?
- If data changes within the tsu/th window it could happen.
- Whenever clock and data have an unknown phase relationship it is impossible for either signal to know when the other is about to change value.
- Thus you cannot guarantee that tsu and th will not be violated with arbitrary clock and data phase.

![Diagram of a D flip-flop with labels for data, clk, reset_n, output, window, tsu, th, and timing windows for 0.5um and 0.18um CMOS.]
Instead of the nice clean transition you might expect, a metastable output of a flip-flop can look like this. A Tektronix 11801 digital sampling scope was used here to characterize an ECL flip-flop.
(from EDN Dec. 10, 1992 "Exorcise Metastability from your Design", p58-64)
Sampling scope photo of flip-flop going metastable. Each dot represents probability of traces that pass through that region. (Thomas J. Chaney, Washington Univ.)

Multiple real-time scope traces of flip-flop going metastable. (Thomas J. Chaney, Washington Univ.)
Is the problem of metastability real or only an academic topic?

- Push button switch, computer keyboard, microwave button, alarm clock

User input is not synchronized to clk
Real switches make real glitches
Neither pay attention to clk!

-Frequency counter

Timebase and input signal come from different sources

Enable and clock have no known phase relationship!
Asynchronous reset circuit to ASIC

WSC VDU

Metastability
How does metastability manifest itself in a circuit?

-The general case is an asynchronous input to a synchronous state machine

-Paths through logic cloud are different
-An async input to one flip-flop may take the long path
-Late arriving async signal will make setup time to one flip-flop and not to the other.

-In general, the following may be observed:
-The state machine may transition to an illegal state. (stuck?!) 
The state machine flip-flops may exhibit a very long clock to Q delay causing the next stage to go metastable or fail.
- Both flip-flops are running at the same clock frequency. i.e., Synchronous system.

- But clk1 and clk2 comes from two different branches of the clock tree

- Suppose clk 1 branch is slightly faster than clk 2

- If FF1 and FF2 are connected with no logic cloud delay between them, FF2 may get new value of FF1 instead of the old value.

- This problem may be solved by using the logic synthesis tool to go back and find such cases. Usually called a "fix hold" operation.
How is metastability characterized?
- The main impact of metastability is on MTBF (mean time between failures)
- MFTB of XX in 10^yy years does not mean that it wont fail for XX in 10^yy years
- it could fail right away
- MTBF is a statistical measure of reliability

- For a single synchronizing flip-flop, the effect of metastability on MTBF is Fc and Fd are not correlated has been empirically determined to be:

$$MTBF = \left[ \frac{1}{F_c \cdot F_d \cdot T_0} \right] \cdot e^{-\left[ \frac{t'}{T} \right]}$$

where:
Fd = Flip flop data frequency in hertz
Fd = clock frequency in hertz
T0 and T are device specific scaling factors
t' = time required before the metastable state is resolved
\(\frac{1}{T}\) = the speed at which the meta condition is being resolved

-Fd is usually not a clock-like signal and is thus usually estimated to be equivalent to some frequency.
-T,To are related to gain bandwidth (GBW) of the flip-flop and the amount of noise present. The GBW is usually determined by small signal analysis of the flip-flop circuit.
-For a state machine, t' becomes t\text{cycle}-(tsu+tckq+tpd)
Calculation of MTBF

MTBF Calculations (Single F/F)

\[ \text{MTBF} = \frac{1}{F_c f_d T_0} \cdot e^{\left(\frac{t'}{t}\right)} \]

Let \( \frac{t}{t'} = k_2 \), \( T_0 = k_1 \)

\[ t' = 10 - 4 - 0.5 = 5.5 \text{ns} \]

\[ \text{MTBF} = \frac{e^{\left(\frac{5.5 \times 10^{-9} \times 1.268 \times 10^{10}}{100 \times 50 \times 1.01 \times 10^{-13}}\right)}}{505} = \frac{1.25 \times 10^{30}}{505} = 2.47 \times 10^{27} \text{sec} = 7.84 \times 10^{19} \text{years} \]

If we remove the logic delay....

MTBF increases to \( 4.091 \times 10^{42} \) sec or \( 1.29 \times 10^{142} \) years !

30 orders of magnitude improvement. Thus keep delay between synchronizer F/F and state machines small.

If we increase clock to 200MHz... (keep 4ns delay)

\[ \text{MTBF} = \frac{e^{\left(\frac{5.5 \times 10^{-9} \times 1.268 \times 10^{10}}{200 \times 50 \times 1.01 \times 10^{-13}}\right)}}{1010} = \frac{566.7}{1010} = 0.56 \text{sec} ! \]

Remove the 4ns tpd....

\[ \text{MTBF} = \frac{e^{\left(\frac{4.5 \times 10^{-9} \times 1.268 \times 10^{10}}{200 \times 50 \times 1.01 \times 10^{-13}}\right)}}{1010} = \frac{3.42 \times 10^{27}}{1010} = 1.07 \times 10^{17} \text{years} \]
Using the MTBF equation, what can we do to improve MTBF?

-Fc: use the slowest reasonable clock to synchronize the data. Divide the synchronizing clock down before sampling the input signal if you can tolerate the delay.

-Fd: synchronize the data at a point where it is changing slowly. Synchronize on bigger chunks of data. For a serial data stream, synchronize on bytes instead of bits.

-T and To are usually fixed by technology choice. If a choice exists however, use the fastest flip-flops possible or those with the smallest sum of (tsu+th).

-Note that T and To are averages taken from very limited samples of parts. These parameters are very sensitive to variations in temperature, voltage, and process.

-t: To allow for the greatest time for meta states to resolve, minimize or eliminate any delay between the synchronizing flip-flop and the next flip-flop.

Tsu and Th for several digital families

Table 1:

<table>
<thead>
<tr>
<th>Logic Family / Technology</th>
<th>tsu</th>
<th>th</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALS74</td>
<td>15</td>
<td>10</td>
</tr>
<tr>
<td>HC74</td>
<td>20</td>
<td>0</td>
</tr>
<tr>
<td>S74</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>AC74</td>
<td>4</td>
<td>0.5</td>
</tr>
<tr>
<td>1um CMOS</td>
<td>1</td>
<td>0.2</td>
</tr>
<tr>
<td>0.5um COMS</td>
<td>0.3</td>
<td>0</td>
</tr>
<tr>
<td>0.18 CMOS</td>
<td>0.1</td>
<td>0</td>
</tr>
</tbody>
</table>
Xilinx Vertex 2 MTBF Data

XC2VP4 Metastable Recovery
~300 MHz Clock, 50 MHz Data

Metastable Progress
2002 vs 1996
~100 MHz Clock, 1 MHz Data
A solution to the problem, the synchronizer
- Asynchronous data is first clocked by a synchronizer flip-flop prior to being applied to internal synchronous logic.

- The intent is that the synchronizer flip-flop will be able to resolve any metastability at its Q output with enough time left so that the logic cloud delay and tsu will be met at the state machine.

- This solution is perfectly suitable for logic that is fast relative to the clock period. Analysis is necessary however.

- A much better solution is the "dual rank" synchronizer that uses two flip-flops. The probability that both flip-flops will undergo metastability is greatly reduced.
**MTBF of Dual-Rank Synchronizer**

**CALCULATING MTBF FOR A 2-STAGE SYNCHRONIZER**

It is difficult to give a formula for the MTBF of a 2-stage synchronizer (Fig A). The failure whose frequency is being calculated is the failure of the second-stage flip-flop to resolve by time $t'$. The clock frequency at the synchronizer flip-flops ($f_c$) and the data-input transition frequency ($f_d$) are known. The difficulty is in determining $f_{a2}$, the number of data-input transitions expected/unit of time for the second flip-flop.

One possible assumption is to let $f_{a2}$ be the probability that the first flip-flop has not settled by one setup time before the clock of the second flip-flop, $(1/f_c - T_{a2})$. Then, the following equation (Ref 4) shows the MTBF for the synchronizer, assuming both have the same metastability parameters:

$$MTBF = \frac{1}{f_{a2} \times f_c \times T_0} \times e^{\frac{-1}{f_c - T_{a2}}}.$$

By assumption, $1/f_{a2}$ = MTBF of the first synchronizer (MTBF$_1$).

$$MTBF = \frac{1}{f_{a2} \times f_c \times T_0} \times e^{\frac{1}{f_c - T_{a2}}}.$$

Therefore,

$$MTBF = \frac{1}{f_d \times f_c \times T_0} \times e^{\frac{t'}{f_c - T_{a2}}} = \frac{1}{f_d \times f_c \times T_0} \times e^{\frac{t'}{f_c \times T_{a2}}}.$$

Because the $f_d$ term appears only once, this is not the square of MTBF$_1$, as is sometimes claimed.

Setting $f_{a2} = 1/MTBF_1$ assumes that one uniformly distributed asynchronous data transition occurs each time the first stage goes metastable. One could argue that this assumption doesn't necessarily hold. The apparent $f_{a2}$ depends on the first flip-flop's metastable behavior. For example, oscillations and intermediate voltage levels from the first flip-flop could more likely to cause setup violations on the second one, producing a larger apparent $f_{a2}$ that would distort pulses and delayed transitions. Nevertheless, errors in $f_{a2}$ are insignificant compared with uncertainties in the exponential term.

**Fig A**—Calculating the MTBF of a 2-stage synchronizer requires an estimate of $f_{a2}$.

112 • EDN June 23, 1994
Conclusion
-Metastability is a fundamental problem
-It occurs in many places
-It cannot be solved, there is no perfect solution
-We cannot guarantee correct operation when clock and data have arbitrary phase relationships.
-All asynchronous interfaces will fail someday if left running

Metastability is "special" because
-it defies most conceptual and computational tools we use
-simulators do not handle it at all
-it defies measurements, only statistical data can be gathered
-it causes failures in hardware and software that are "perfect"
-it causes failures that often leave no "smoking gun"
-it is very difficult to test for in any conventional sense
-it involves time and voltage magnitudes far beyond our usual experience

Guidelines
-Be careful! Carefully figure MTBF.
-Watch out where multiple identical parts have asynchronous interfaces.
-Minimize the number of asynchronous interfaces
-Synchronize with slowest clock and data
-Immediately synchronize async inputs
-Asynchronous inputs only go to one flip-flop, the synchronizer flip-flop
-Use fastest available technology/cells.