

In this second guest lecture we will discuss why timing constraints in synchronous designs are important and what can happen if they are violated.

HWMoD
WS25

Metastability
Recap
Metastability

Hardware Modeling [VU] (191.011) – WS25 – Metastability

Guest Lecture by Prof. Steininger

WS 2025/26

As it has been quite some time between the previous guest lecture and this one, let us briefly recapitulate the basic idea of the synchronous design style at the beginning. Hardware is operating highly parallel due to it essentially being a complex network of a myriad of concurrently working combinational gates. These gates immediately react to input changes by mapping it to a new output value that is stable after their propagation delay. In a typical circuit gates are concatenated, meaning that some gates provide the inputs for others. Since signals may be subject to different delays along different paths, the operation of such a circuit easily becomes a mess.

Recap: Synchronous Design

HWMoD
WS25

Metastability
Recap
Metastability

- Hardware usually operates with high concurrency
 - Circuits consist of complex networks of comb. gates
 - Combinational gates immediately react to input changes

- Hardware usually operates with high concurrency
 - Circuits consist of complex networks of comb. gates
 - Combinational gates immediately react to input changes
- Coordination is required for proper operation
 - Inputs must be stable throughout computation
 - Outputs must be valid when used

To mitigate this, coordination that ensures a proper operation of the circuit is required. This coordination has to ensure that inputs remain stable while they are processed, and that outputs are only used once they are stable.

Recap: Synchronous Design

HWMoD
WS25

Metastability
Recap
Metastability

- Hardware usually operates with high concurrency
 - Circuits consist of complex networks of comb. gates
 - Combinational gates immediately react to input changes
- Coordination is required for proper operation
 - Inputs must be stable throughout computation
 - Outputs must be valid when used

- Hardware usually operates with high concurrency
 - Circuits consist of complex networks of comb. gates
 - Combinational gates immediately react to input changes
- Coordination is required for proper operation
 - Inputs must be stable throughout computation
 - Outputs must be valid when used
- ⇒ Use a global clock signal as common notion of time
 - Flip-flops between combinational logic control signal propagation

In the synchronous design style this coordinated is provided by a central clock signal which provides a global time base to which all function blocks can align their activities. However, since combinational gates are insensitive to the clock by design, we must introduce sequential circuit elements that capture and pass signals in a controlled manner. In the lecture about the synchronous design style we introduced flip-flops for that purpose.

Recap: Synchronous Design

HWMoD
WS25

Metastability
Recap
Metastability

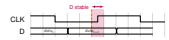
- Hardware usually operates with high concurrency
 - Circuits consist of complex networks of comb. gates
 - Combinational gates immediately react to input changes
- Coordination is required for proper operation
 - Inputs must be stable throughout computation
 - Outputs must be valid when used
- ⇒ Use a global clock signal as common notion of time
 - Flip-flops between combinational logic control signal propagation

- Hardware usually operates with high concurrency
 - Circuits consist of complex networks of comb. gates
 - Combinational gates immediately react to input changes
- Coordination is required for proper operation
 - Inputs must be stable throughout computation
 - Outputs must be valid when used
- ⇒ Use a global clock signal as common notion of time
 - Flip-flops between combinational logic control signal propagation
 - Flip-flops have inherent timing constraints

However, as we mentioned in that lecture, these flip-flops also have timing constraints that must be satisfied for a proper operation. We will look into this matter in more depth in this lecture.

Recap: Synchronous Design

- Hardware usually operates with high concurrency
 - Circuits consist of complex networks of comb. gates
 - Combinational gates immediately react to input changes
 - Coordination is required for proper operation
 - Inputs must be stable throughout computation
 - Outputs must be valid when used
- ⇒ Use a global clock signal as common notion of time
- Flip-flops between combinational logic control signal propagation
 - Flip-flops have inherent timing constraints



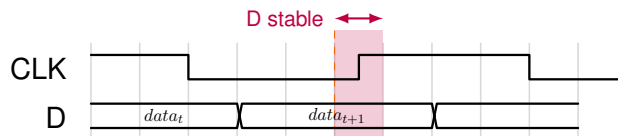
In the previous guest lecture we already briefly mentioned that the most important timing condition for a flip-flop is that its input must not change around the active clock edge where it is sampled. If the input changes around the clock edge, it is not clear which value the flip should capture. Let us now break down the times around a clock edge in which the data must be stable and discuss why they exist.

Flip-Flop Timing Constraints

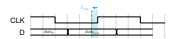
HWMoD
WS25

Metastability
Recap
Metastability

- Input data must be stable around active clock edge
 - Otherwise not clear which value to capture



- Input data must be stable around active clock edge
- Otherwise not clear which value to capture
- Setup Time: specifies how long data must be stable before clock edge



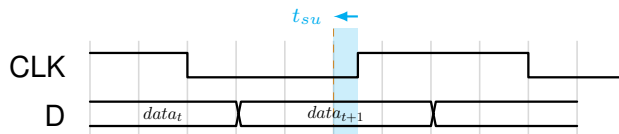
The so-called *setup* time of a flip-flop specifies how long the input data must be stable *before* the active clock edge. During this time, the signal propagates and then stabilizes at the internal part of the flip-flop where it is then actually sampled at the clock edge. Thus, this timing constraint essentially ensures that the data is internally stable when the copying starts.

Flip-Flop Timing Constraints

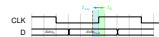
HWMoD
WS25

Metastability
Recap
Metastability

- Input data must be stable around active clock edge
 - Otherwise not clear which value to capture
 - Setup Time:** specifies how long data must be stable *before* clock edge



- Input data must be stable around active clock edge
- Otherwise not clear which value to capture
- Setup Time:** specifies how long data must be stable before clock edge
- Hold Time:** specifies how long data must be stable after clock edge



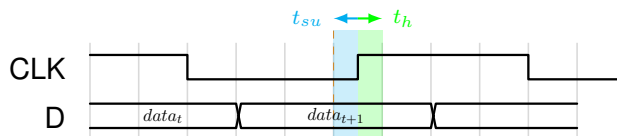
Conversely, the so-called *hold* time specifies how long the input data must be stable *after* the active clock edge. The reason for this timing constraint is that the flip-flop cannot capture its input instantaneously, but rather this takes some time. If the input changes during this time, a wrong value could be captured.

Flip-Flop Timing Constraints

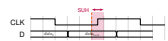
HWMoD
WS25

Metastability
Recap
Metastability

- Input data must be stable around active clock edge
 - Otherwise not clear which value to capture
 - Setup Time:** specifies how long data must be stable *before* clock edge
 - Hold Time:** specifies how long data must be stable *after* clock edge



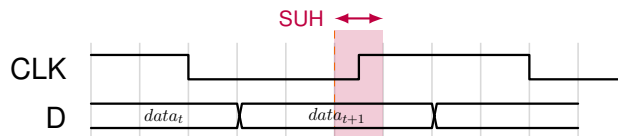
- Input data must be stable around active clock edge
- Otherwise not clear which value to capture
- Setup Time:** specifies how long data must be stable before clock edge
- Hold Time:** specifies how long data must be stable after clock edge
- This is the **setup-hold window (SUH window)**



In total, as you already heard, we refer to this window as the *setup-hold* window.

Flip-Flop Timing Constraints

- Input data must be stable around active clock edge
 - Otherwise not clear which value to capture
 - Setup Time:** specifies how long data must be stable *before* clock edge
 - Hold Time:** specifies how long data must be stable *after* clock edge
- This is the **setup-hold window** (SUH window)



Recall that the task of the so-called static timing analysis automatically checks for timing violations in a circuit and warns designers about them. It does that by validating that the clock period is long enough to facilitate the arrival of input data at all inputs before the setup window, and that any propagation delay between two flip-flops is longer than the hold time of the capturing flip-flop. Thus, everything is well-checked and should work without problems.

Timing Violations?

- Static timing analysis ensures sufficiently long clock period for all timing constraints of FFs and comb. logic to be satisfied

Are timing violations thus even possible when making use of the static timing analysis and paying heed to its warnings? Well, to some extent yes. However, as you might already guess given that we discuss this topic, that's not the whole story.

Timing Violations?

- Static timing analysis ensures sufficiently long clock period for all timing constraints of FFs and comb. logic to be satisfied
- ⇒ Are timing violations then even possible? Why bother?

Obviously, every useful circuit requires some interface to the world it resides in. However, the issue is that at these interfaces we leave the synchronous abstraction we establish within our circuit, as the outside world does not know about an internal clock.

Timing Violations?

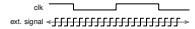
- Static timing analysis ensures sufficiently long clock period for all timing constraints of FFs and comb. logic to be satisfied
- ⇒ Are timing violations then even possible? Why bother?
- Every useful circuit requires an interface to the outside world

Metastability

Recap

Timing Violations?

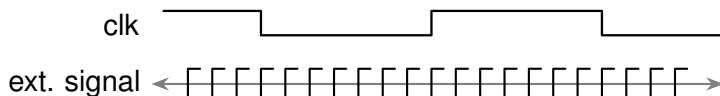
- Static timing analysis ensures sufficiently long clock period for all timing constraints of FFs and comb. logic to be satisfied
- ⇒ Are timing violations then even possible? Why bother?
- Every useful circuit requires an interface to the outside world
 - Transition at external inputs will arrive at *any* time



As a result, transitions on external inputs while happen at *any* time with respect to the clock. Naturally, this means that the outside world cannot respect the timing conditions of our circuit which can lead to timing violations. The timing diagram on the slide captures this, where the upper signal, called clk is a synchronous design's internal clock signal and below a completely uncorrelated external signal is illustrated.

Timing Violations?

- Static timing analysis ensures sufficiently long clock period for all timing constraints of FFs and comb. logic to be satisfied
- ⇒ Are timing violations then even possible? Why bother?
- Every useful circuit requires an interface to the outside world
 - Transition at external inputs will arrive at *any* time

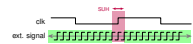


Metastability

Recap

Timing Violations?

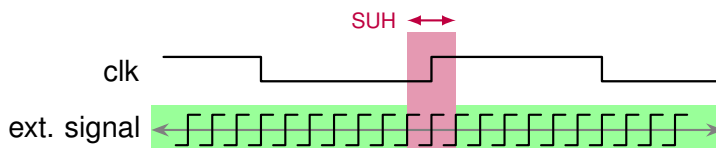
- Static timing analysis ensures sufficiently long clock period for all timing constraints of FFs and comb. logic to be satisfied
- ⇒ Are timing violations then even possible? Why bother?
- Every useful circuit requires an interface to the outside world
- Transition at external inputs will arrive at any time
- In particular: they can arrive within a SUH window



In particular, such transitions can happen within the setup-hold window of a flip-flop directly sampling this input or the result of a combinational function that depends on this input. However, note that this is really something that *can* happen for any input transition but does not necessarily have to. On the slide we highlight this by marking the time windows where input transitions will not cause any problems in green, and the problematic setup-hold window in red. To realize that this really is a problem, just imagine that you have to push a button that is connected to a system with a clock frequency of a couple of hundred megahertz and that you must not hit a setup-hold window. This is also not just some theoretical thought experiment, but the origin of real computers misbehaving. Already decades in the past, designers already used the static timing analysis but discovered the computers they built to misbehave at seemingly random moments when inputs, for example buttons, changed.

Timing Violations?

- Static timing analysis ensures sufficiently long clock period for all timing constraints of FFs and comb. logic to be satisfied
- ⇒ Are timing violations then even possible? Why bother?
- Every useful circuit requires an interface to the outside world
 - Transition at external inputs will arrive at *any* time
 - In particular: they can arrive within a SUH window



Now that we know that timing violations are still possible, even when using a static timing analysis, let us discuss the consequences of them.

Timing Violations!

HWMoD
WS25

Metastability

Recap

Metastability

Analogy

Consequences

MTBU

- What happens *if* the timing constraints are violated?

First, we have to distinguish between combinational gates and functions, and flip-flops.

Timing Violations!

HWMoD
WS25

Metastability

Recap

Metastability

Analogy

Consequences

MTBU

- What happens *if* the timing constraints are violated?
- Distinguish between combinational gates and sequential FFs

Combinational gates are in principle very resilient to the unpredictable nature of external input transitions. They continuously process their inputs, no matter when transitions arrive, and they produce a respective output if the input was stable for long enough.

As introduced in the lecture about the synchronous design style, for them to work correctly, we need to ensure that their result is only needed after the time it takes for the input signals to completely propagate through the combinational logic. In synchronous designs this is ensured by the clock period being longer than any such time. In the worst case, if its input was not stable for long enough, combinational logic will produce incomplete results, where only some outputs have changed.

Timing Violations!

HWMoD
WS25

Metastability
Recap
Metastability
Analogy
Consequences
MTBU

- What happens *if* the timing constraints are violated?
- Distinguish between combinational gates and sequential FFs
 - Comb. gates: simply produce incomplete results

For flip-flops the consequences are more severe. As we already mentioned, external transitions will occasionally violate the setup and hold time of their sampling flip-flop. As we already suggested, this results in the fact that it is not clear for the flip-flop which value to capture and forward. However, what exactly does this mean? As it turns out, violating timing conditions of flip-flops can lead to them becoming *metastable*. We will now discuss what this means.

Timing Violations!

HWMoD
WS25

Metastability
Recap
Metastability
Analogy
Consequences
MTBU

- What happens *if* the timing constraints are violated?
- Distinguish between combinational gates and sequential FFs
 - Comb. gates: simply produce incomplete results
 - Flip-flop: **Metastability**

- Metastability
- Metastability
- Flip-Flop Metastability**

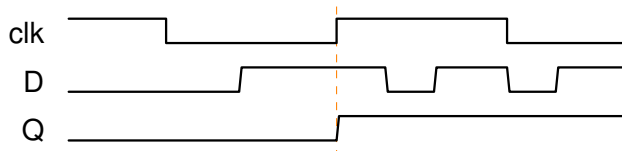
- The flip-flop is supposed to assume one of two states after a clock edge
- State reflected by the output



Recall that the function of a flip-flop is to sample the logic level of its data input at each active clock edge, and to provide this level at its output as shown on the slide. Between active clock edges, it has to keep its output value stable, no matter whether the input changes. As you already heard, this means that a flip-flop must store the value it sampled internally. We also refer to this as the state of a flip-flop.

Flip-Flop Metastability

- The flip-flop is supposed to assume one of two states after a clock edge
 - State reflected by the output



HWMoD
WS25

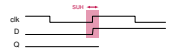
Metastability
Recap
Metastability
Analogy
Consequences
MTBU

Metastability

Metastability

Flip-Flop Metastability

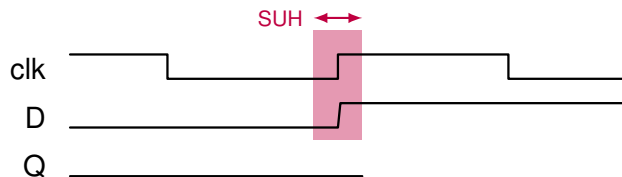
- The flip-flop is supposed to assume one of two states after a clock edge
 - State reflected by the output
 - Transition during SUH window ⇒ FF might not be able to decide on state



Let us now assume that the input of flip-flop changes during its setup-hold window as shown on the slide. As a result, the flip-flop might not be able to decide whether it should capture and output low or high. As you will hear in other courses, since low and high are the only stable states of a flip flop, it *has* to decide for one. However, which should it take if a voltage exactly in the middle of low and high is applied? Well, in the worst case it cannot.

Flip-Flop Metastability

- The flip-flop is supposed to assume one of two states after a clock edge
 - State reflected by the output
- Transition during SUH window ⇒ FF might not be able to decide on state

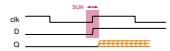


HWMoD
WS25

Metastability
Recap
Metastability
Analogy
Consequences
MTBU

- └─ Metastability
- └─ Metastability
- └─ **Flip-Flop Metastability**

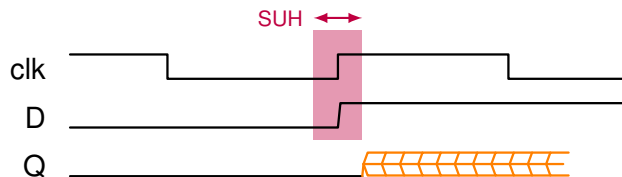
- The flip-flop is supposed to assume one of two states after a clock edge
 - State reflected by the output
- Transition during SUH window ⇒ FF might not be able to decide on state
- The FF is *metastable* (i.e., between two stable states)



We refer to this state, as the flip-flop being metastable. While being metastable, the flip-flop is deciding whether its state will be low or high. This is illustrated on the slide where the output Q remains at some intermediate voltage between low and high and where it can either change to low or high at an arbitrary point in time. Note that during that phase it will also deliver an intermediate voltage level at its output.

Flip-Flop Metastability

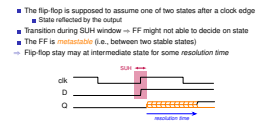
- The flip-flop is supposed to assume one of two states after a clock edge
 - State reflected by the output
- Transition during SUH window ⇒ FF might not be able to decide on state
- The FF is *metastable* (i.e., between two stable states)



Metastability

Metastability

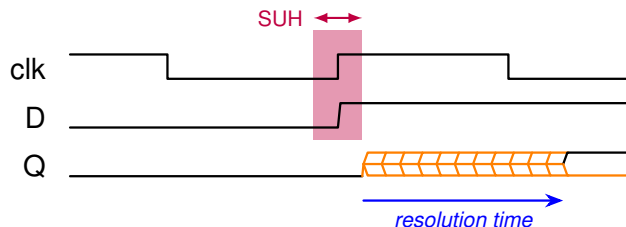
Flip-Flop Metastability



A flip-flop in this state will stay undecided for an arbitrary long time. However, as the metastable state is not a stable state itself, even the slightest nudge will tip the flip-flop to either state. Afterwards, it will be in one of its two stable states, delivering a valid digital voltage value at its output. On the slide we have arbitrarily chosen the flip-flop to decide to transition to its high state. Note that we refer to the time between the active clock edge and flip-flop tipping as the *resolution time*.

Flip-Flop Metastability

- The flip-flop is supposed to assume one of two states after a clock edge
 - State reflected by the output
 - Transition during SUH window ⇒ FF might not be able to decide on state
 - The FF is *metastable* (i.e., between two stable states)
- ⇒ Flip-flop stay may at intermediate state for some *resolution time*



Metastability

Metastability

Physical Analogy

■ Inherent to any system with transitions between multiple stable states

Granted, the concept of metastability is quite alien, especially if you have not heard of it before. Therefore, let us consider some physical analogies. In essence, any system that transitions between stable states is subject to metastability, as there will be certain point between stable states where both states are equally likely to be the outcome of the transition. If this point is reached, the system thus becomes metastable.

Physical Analogy

- Inherent to any system with transitions between multiple stable states

HWMMod
WS25

Metastability

Recap

Metastability

Analogy

Consequences

MTBU

Metastability

Metastability

Physical Analogy

- Inherent to any system with transitions between multiple stable states
- Metastability is the act of *balancing* between stable states
- Output voltage of FF, elephant on ball...



To make this concept a bit more tangible, we can view metastability to be the act of balancing between stable states. A flip-flop balancing around the voltage exactly between low and high is just one instance of this phenomenon. For example, consider the image of the elephant balancing on the ball. As we all intuitively know, the elephant will eventually fall off the ball. However, we cannot predict when exactly this will happen and to which side it will fall. We can not even give a bound for when the elephant will off, as in theory it could even stay atop the ball forever. In reality though, we know that this is extremely unlikely. In fact, the longer the elephant stays atop, the unlikelier it becomes for it to remain there.

Physical Analogy

- Inherent to any system with transitions between multiple stable states
- Metastability is the act of *balancing* between stable states
 - Output voltage of FF, elephant on ball...



HWMMod
WS25

Metastability

Recap

Metastability

Analogy

Consequences

MTBU

Metastability

Metastability

Physical Analogy

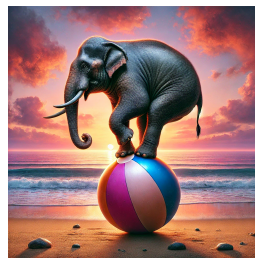
- Inherent to any system with transitions between multiple stable states
- Metastability is the act of *balancing* between stable states
- Output voltage of FF, elephant on ball...
- In general: Metastability **cannot** be mitigated!



An important fact about metastability is that it can in general not be avoided, as it is a direct consequence of transitions between stable states. In fact, there even exist formal proofs that show this.

Physical Analogy

- Inherent to any system with transitions between multiple stable states
- Metastability is the act of *balancing* between stable states
 - Output voltage of FF, elephant on ball...
- In general: Metastability **cannot** be mitigated!



HWMMod
WS25

Metastability

Recap

Metastability

Analogy

Consequences

MTBU

Metastability

Metastability

Physical Analogy

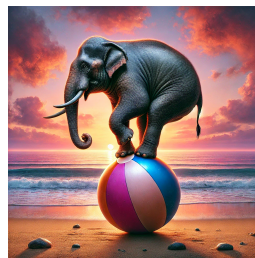
- Inherent to any system with transitions between multiple stable states
- Metastability is the act of *balancing* between stable states
 - Output voltage of FF, elephant on ball...
- In general: Metastability **cannot** be mitigated!
 - Neither resolution time nor final outcome can be determined in advance



In particular, neither the time it takes for the system to resolve to a stable state, nor the final outcome of this resolution process can be determined in advance. Intuitively this quite tangible. If this could be determined, metastability could be avoided, which is impossible. Let us now discuss why the metastability of a flip-flop is undesired.

Physical Analogy

- Inherent to any system with transitions between multiple stable states
- Metastability is the act of *balancing* between stable states
 - Output voltage of FF, elephant on ball...
- In general: Metastability **cannot** be mitigated!
 - Neither resolution time nor final outcome can be determined in advance

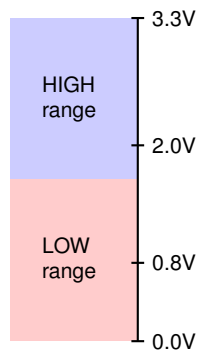




For that, let us first understand how a receiver of a logic signal determines its logic level. If we consider binary logic, a receiver must essentially discretize its analog input voltages into two discrete levels, referred to as low and high, respectively zero and one.

Consequences of Metastability in Circuits

- Obtaining binary logic levels by discretizing analog voltage



Metastability

Metastability

Consequences of Metastability in Circuits

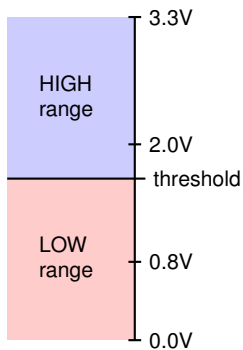
- Obtaining binary logic levels by discretizing analog voltage
- Comparison against threshold voltage



Basically, circuit elements do this by comparing their input voltages with an internal threshold. Ideally the threshold voltage is around half the supply voltage. Input voltages above this threshold will be considered to be high, and the other voltages to be low.

Consequences of Metastability in Circuits

- Obtaining binary logic levels by discretizing analog voltage
 - Comparison against threshold voltage



HWMMod
WS25

Metastability
Recap
Metastability
Analogy
Consequences
MTBU

Metastability

Metastability

Consequences of Metastability in Circuits

■ Obtaining binary logic levels by discretizing analog voltage

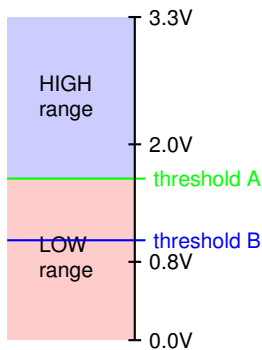
■ Comparison against threshold voltage



However, due to fabrication tolerances, this threshold voltage varies between different circuit elements of the same technology, as illustrated on the slide.

Consequences of Metastability in Circuits

- Obtaining binary logic levels by discretizing analog voltage
 - Comparison against threshold voltage



HWMMod
WS25

Metastability
Recap
Metastability
Analogy
Consequences
MTBU

Metastability

Metastability

Consequences of Metastability in Circuits

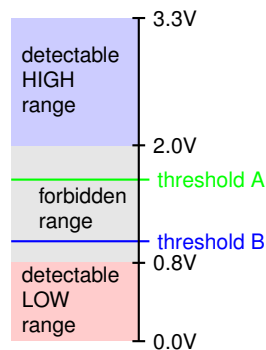
- Obtaining binary logic levels by discretizing analog voltage
- Comparison against threshold voltage



To account for these tolerances, input voltages are restricted to certain voltage ranges that will, under all circumstance, be detected as the same logical level by all parts of the circuit. Note that there is a range in which all threshold voltages lie. Voltage levels in this range may be categorized differently by parts of the circuit. To avoid this, voltages in this range are forbidden, meaning no circuit element must produce an output within this range. As a result of this, the tolerances in the threshold voltages do not matter during the regular operation of a circuit.

Consequences of Metastability in Circuits

- Obtaining binary logic levels by discretizing analog voltage
 - Comparison against threshold voltage



HWMMod
WS25

Metastability
Recap
Metastability
Analogy
Consequences
MTBU

Metastability

Metastability

Consequences of Metastability in Circuits

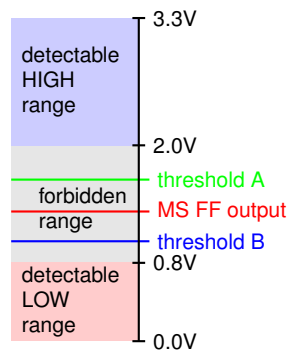
- Obtaining binary logic levels by discretizing analog voltage
- Comparison against threshold voltage
- Metastable flip-flop outputs intermediate voltage



However, as we already heard, a metastable flip-flop might produce an intermediate voltage level around half the supply voltage. This voltage can be within the forbidden voltage range.

Consequences of Metastability in Circuits

- Obtaining binary logic levels by discretizing analog voltage
 - Comparison against threshold voltage
- Metastable flip-flop outputs intermediate voltage



HWMMod
WS25

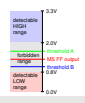
Metastability
Recap
Metastability
Analogy
Consequences
MTBU

Metastability

Metastability

Consequences of Metastability in Circuits

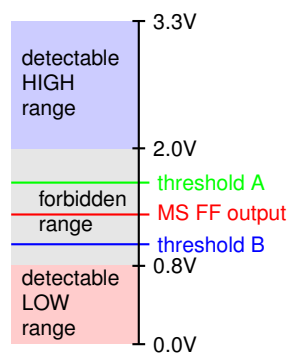
- Obtaining binary logic levels by discretizing analog voltage
 - Comparison against threshold voltage
- Metastable flip-flop outputs intermediate voltage
- Depending on particular threshold voltages different interpretation



Thus, depending on the threshold voltages, the output of a metastable flip-flop might be interpreted differently by different parts of the circuit. For example, consider the voltage thresholds of a flip-flop A and B as shown on the slide. In this case, flip-flop A will observe a low input while flip-flop B will observe a high one. This different interpretation can of course to undesired or erroneous behavior.

Consequences of Metastability in Circuits

- Obtaining binary logic levels by discretizing analog voltage
 - Comparison against threshold voltage
- Metastable flip-flop outputs intermediate voltage
- Depending on particular threshold voltages different interpretation



HWMMod
WS25

Metastability
Recap
Metastability
Analogy
Consequences
MTBU

Metastability

Metastability

Consequences of Metastability in Circuits

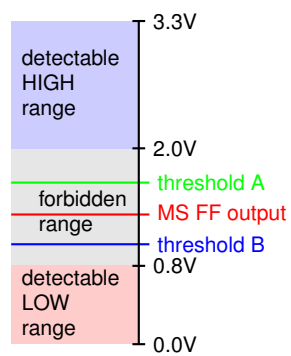
- Obtaining binary logic levels by discretizing analog voltage
 - Comparison against threshold voltage
- Metastable flip-flop outputs intermediate voltage
- Depending on particular threshold voltages different interpretation
- Late transitions, Glitches, Oscillations



Another possible effect is that the output voltage of a metastable flip-flop does not cross the threshold of some other circuit element while transitioning to the intermediate value, but only later when it resolved. This leads to a late transition at the input of the next element, possibly violating its timing constraints. Alternatively, the voltage trace may already cross the threshold initially, but then the metastability resolution will bring the voltage back to its initial value, causing another threshold crossing. In this case we see an undesired pulse to which we refer to as glitch. Finally, under certain conditions, metastability may even lead to the oscillation of a signal. Needless to say, all these effects are undesired and potentially problematic.

Consequences of Metastability in Circuits

- Obtaining binary logic levels by discretizing analog voltage
 - Comparison against threshold voltage
- Metastable flip-flop outputs intermediate voltage
- Depending on particular threshold voltages different interpretation
- Late transitions, Glitches, Oscillations



HWMoD
WS25

Metastability
Recap
Metastability
Analogy
Consequences
MTBU

- └ Metastability
 - └ Metastability
 - └ **Estimating Effects of Metastability**

■ MS cannot be mitigated, can we determine how often it causes problems?

As we heard before, metastability is in general unavoidable, and we thus have to live with it. Therefore, important information would be how often it will occur and, in particular, how often it will cause a problem.

Estimating Effects of Metastability

- MS cannot be mitigated, can we determine how often it causes problems?

HWMoD
WS25

Metastability
Recap
Metastability
Analogy
Consequences
MTBU

Metastability

Metastability

Estimating Effects of Metastability

- MS cannot be mitigated, can we determine how often it causes problems?
- Two contributing factors:
 - How often input transitions fall within SH window
 - How often metastability resolves before propagating

In principle, there are two factors we need to consider then characterizing metastability. The first one is how often a setup-hold window is even violated. If this is happening rarely, the risk of a flip-flop becoming metastable and related problems is lower than when a setup-hold window is violated all the time. In particular, we are interested in the probability of an input transition falling into the setup-hold window of a flip-flop during a clock cycle. The second factor is how often the metastability of a flip-flop resolved before its output gets captured, as a flip-flop becoming metastable without any other part of the circuit observing it is in general unproblematic.

Estimating Effects of Metastability

- MS cannot be mitigated, can we determine how often it causes problems?
- Two contributing factors:
 - 1 How often input transitions fall within SH window
 - 2 How often metastability resolves before propagating

HWMoD
WS25

Metastability
Recap
Metastability
Analogy
Consequences
MTBU

Metastability

Metastability

Estimating Effects of Metastability

- MS cannot be mitigated, can we determine how often it causes problems?
- Two contributing factors:
 - How often input transitions fall within SH window
 - How often metastability resolves before propagating
- Input transition rate in general uncorrelated to clock \Rightarrow assume uniform distribution of clock-to-data time

As we already mentioned before, the input transition rate is in general completely uncorrelated to an internal clock. Therefore, we have to assume that any time between a data transition and an active clock edge is equally probable. As a result, we assume a uniform distribution for this time and thus also for the times of the input transitions.

Estimating Effects of Metastability

HWMMod
WS25

Metastability
Recap
Metastability
Analogy
Consequences
MTBU

- MS cannot be mitigated, can we determine how often it causes problems?
- Two contributing factors:
 - 1 How often input transitions fall within SH window
 - 2 How often metastability resolves before propagating
- Input transition rate in general uncorrelated to clock \Rightarrow assume uniform distribution of clock-to-data time

Metastability

Metastability

Estimating Effects of Metastability

- MS cannot be mitigated, can we determine how often it causes problems?
- Two contributing factors:
 - How often input transitions fall within SH window
 - How often metastability resolves before propagating
- Input transition rate in general uncorrelated to clock \Rightarrow assume uniform distribution of clock-to-data time
- Resolution time not predictable \Rightarrow only statistical estimation possible

Furthermore, we already mentioned before that the resolution time is not predictable. In conclusion, we already know that the best we can in general get is a statistical characterization of metastability.

Estimating Effects of Metastability

HWMMod
WS25

Metastability
Recap
Metastability
Analogy
Consequences
MTBU

- MS cannot be mitigated, can we determine how often it causes problems?
- Two contributing factors:
 - 1 How often input transitions fall within SH window
 - 2 How often metastability resolves before propagating
- Input transition rate in general uncorrelated to clock \Rightarrow assume uniform distribution of clock-to-data time
- Resolution time not predictable \Rightarrow only statistical estimation possible

Metastability

Metastability

Estimating Effects of Metastability

- MS cannot be mitigated, can we determine how often it causes problems?
 - Two contributing factors:
 - How often input transitions fall within SH window
 - How often metastability resolves before propagating
 - Input transition rate in general uncorrelated to clock \Rightarrow assume uniform distribution of clock-to-data time
 - Resolution time not predictable \Rightarrow only statistical estimation possible
- \Rightarrow Mean Time Between Upsets (MTBU)

$$MTBU = \frac{1}{\lambda_{in} \cdot f_{clk} \cdot T_W} \cdot e^{\frac{t_{res}}{\tau_C}}$$

Most commonly, this statistical characterization is expressed via the so-called *Mean Time Between Upsets*, or MTBU for short. This parameter is the expected time that a system will run without a problem since the last upset occurred. At this point, we should note that we talk about an upset whenever the metastable state of a flip-flop has not resolved before its output is being captured. Note that a high MTBU value is desirable as it translates to less frequent upsets. On the slide you can see the widely used equation for estimating this parameter. We will now briefly discuss it such that you can get an intuition on what it expresses.

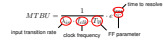
Estimating Effects of Metastability

- MS cannot be mitigated, can we determine how often it causes problems?
 - Two contributing factors:
 - How often input transitions fall within SH window
 - How often metastability resolves before propagating
 - Input transition rate in general uncorrelated to clock \Rightarrow assume uniform distribution of clock-to-data time
 - Resolution time not predictable \Rightarrow only statistical estimation possible
- \Rightarrow *Mean Time Between Upsets* (MTBU)

$$MTBU = \frac{1}{\lambda_{in} \cdot f_{clk} \cdot T_W} \cdot e^{\frac{t_{res}}{\tau_C}}$$

- └ Metastability
- └ Metastability
- └ **MTBU Estimation**

■ MTBU depends on technology and circuit parameters



The first thing to note is the different parameters occurring in the formula. All of them are either technology parameters of the sampling flip-flop that depends on its particular implementation and fabrication process, or parameters of the circuit which define how the flip-flop is operated.

MTBU Estimation

- MTBU depends on technology and circuit parameters

$$MTBU = \frac{1}{\lambda_{in} \cdot f_{clk} \cdot T_W} \cdot e^{\frac{t_{res}}{\tau_C}}$$

input transition rate
clock frequency
FF parameter

time to resolve

HWMMod
WS25

Metastability
Recap
Metastability
Analogy
Consequences
MTBU



While we are not going to explain the technology parameters in detail, we want to point at some particularities of the other parameters. First, note that while we use a frequency for the clock, we use a transition rate for the input. This is because either transition of the input could be problematic with respect to falling within a setup-hold window and because nothing is known about the input signal. In particular, we do not even know if it is periodic. Second, the time to resolve has an exponential influence on the MTBU. This parameter expresses how much time there is between the arrival of the output at the next flip-flop and the point in time where it must be stable. Thus, the more time we give a flip-flop to resolve, the less likely it becomes that a following flip-flop captures a problematic value.

MTBU Estimation

- MTBU depends on technology and circuit parameters
 - Note: **Rate** of input transitions
 - Exponential influence of t_{res} on MTBU!

$$MTBU = \frac{1}{\lambda_{in} \cdot f_{clk} \cdot T_W} \cdot e^{\frac{t_{res}}{\tau_C}}$$

input transition rate clock frequency FF parameter time to resolve

$$MTBU = \frac{1}{\lambda_{in} \cdot f_{clk} \cdot T_W} \cdot e^{\frac{t_{res}}{\tau_C}} \quad \frac{1}{MTBU} = UR = \lambda_{in} \cdot \frac{T_W}{T_{clk}} \cdot e^{-\frac{t_{res}}{\tau_C}}$$

Let us now briefly discuss how this formula is constructed. For that, we invert the MTBU and obtain the rate of upsets. In this inverted form, we can map the two factors we previously mentioned directly to the formula.

MTBU Estimation

- MTBU depends on technology and circuit parameters
 - Note: **Rate** of input transitions
 - Exponential influence of t_{res} on MTBU!
- Where does this formula come from? Consider upset rate (UR)

$$MTBU = \frac{1}{\lambda_{in} \cdot f_{clk} \cdot T_W} \cdot e^{\frac{t_{res}}{\tau_C}}$$

$$\frac{1}{MTBU} = UR = \lambda_{in} \cdot \frac{T_W}{T_{clk}} \cdot e^{-\frac{t_{res}}{\tau_C}}$$

First, we had a factor that expresses that an input transition happens during a setup-hold window. This is essentially the amount of transitions times the odds of hitting a setup-hold window.

MTBU Estimation

- MTBU depends on technology and circuit parameters
 - Note: **Rate** of input transitions
 - Exponential influence of t_{res} on MTBU!
- Where does this formula come from? Consider upset rate (UR)
 - 1 How often input transitions fall within SH window

$$MTBU = \frac{1}{\lambda_{in} \cdot f_{clk} \cdot T_W} \cdot e^{\frac{t_{res}}{\tau_C}}$$

$$\frac{1}{MTBU} = UR = \lambda_{in} \cdot \frac{T_W}{T_{clk}} \cdot e^{-\frac{t_{res}}{\tau_C}}$$

avg. rate of transitions within SH window

- MTBU depends on technology and circuit parameters
 - Note: **Rate** of input transitions
 - Exponential influence of t_{res} on MTBU!
- Where does this formula come from? Consider upset rate (UR)
 - How often input transitions fall within SH window
 - How often metastability resolves before propagating

proportion of MS cases not resolving in time

$$MTBU = \frac{1}{\lambda_{in} \cdot f_{clk} \cdot T_W} \cdot e^{\frac{t_{res}}{\tau_C}}$$

avg. rate of transitions within SH window

Second, we had a factor that expresses that not all occurrences of metastability lead to an upset. It may happen that a flip-flop resolves early enough for the next flip-flop to not be affected, which is captured by the highlighted exponential term.

MTBU Estimation

- MTBU depends on technology and circuit parameters
 - Note: **Rate** of input transitions
 - Exponential influence of t_{res} on MTBU!
- Where does this formula come from? Consider upset rate (UR)
 - How often input transitions fall within SH window
 - How often metastability resolves before propagating

proportion of MS cases not resolving in time

$$MTBU = \frac{1}{\lambda_{in} \cdot f_{clk} \cdot T_W} \cdot e^{\frac{t_{res}}{\tau_C}}$$

$$\frac{1}{MTBU} = UR = \lambda_{in} \cdot \frac{T_W}{T_{clk}} \cdot e^{-\frac{t_{res}}{\tau_C}}$$

avg. rate of transitions within SH window

- MTBU depends on technology and circuit parameters
 - Note: **Rate** of input transitions
 - Exponential influence of t_{res} on MTBU!
- Where does this formula come from? Consider upset rate (UR)
 - How often input transitions fall within SH window
 - How often metastability resolves before propagating
- Formula applicable for *uncorrelated* input data **only!**

proportion of MS cases not resolving in time

$$MTBU = \frac{1}{\lambda_{in} \cdot f_{clk} \cdot T_W} \cdot e^{\frac{t_{res}}{\tau_C}} = UR = \lambda_{in} \cdot \frac{T_W}{T_{clk}} \cdot e^{-\frac{t_{res}}{\tau_C}}$$

avg. rate of transitions within SH window

At this point we want to point out two consequences of metastability which this formula tells us. First, increasing the data rate or the clock frequency decreases the MTBU, thus increasing the likelihood of metastability causing problems. Second, increasing the available time for resolution is a very efficient means to obtain a low upset rate and hence a high MTBU. Finally, note that the topics of this lecture will be covered in more depth in other courses and that this was just a brief introduction. So do not fall into despair if some things did not completely make sense yet or if you still have open questions.

MTBU Estimation

- MTBU depends on technology and circuit parameters
 - Note: **Rate** of input transitions
 - Exponential influence of t_{res} on MTBU!
- Where does this formula come from? Consider upset rate (UR)
 - How often input transitions fall within SH window
 - How often metastability resolves before propagating
- Formula applicable for *uncorrelated* input data **only!**

$$MTBU = \frac{1}{\lambda_{in} \cdot f_{clk} \cdot T_W} \cdot e^{\frac{t_{res}}{\tau_C}}$$

proportion of MS cases not resolving in time

$$\frac{1}{MTBU} = UR = \lambda_{in} \cdot \frac{T_W}{T_{clk}} \cdot e^{-\frac{t_{res}}{\tau_C}}$$

avg. rate of transitions within SH window

Thank you for listening! We recommend you to immediately take the self-check test in TUWEL, to see if you understood the material presented in this lecture.

HWMoD
WS25

Metastability

Recap

Metastability

Analogy

Consequences

MTBU

Lecture Complete!