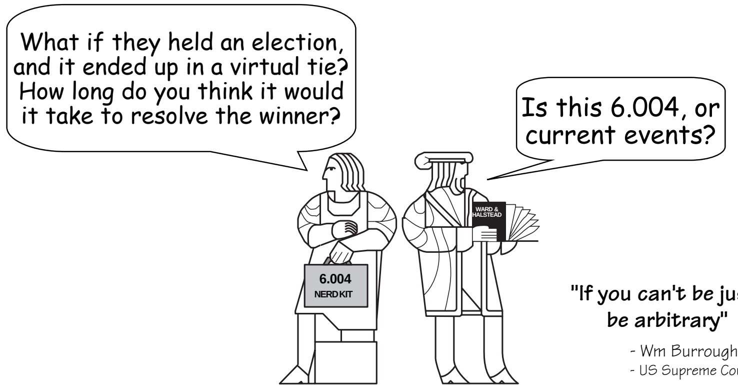
Synchronization, Metastability and Arbitration

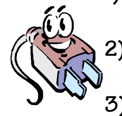

What if they held an election, and it ended up in a virtual tie? How long do you think it would it take to resolve the winner?
Is this 6.004, or current events?
WARD & HALSTEAD
6.004 NERD KIT

"If you can't be just, be arbitrary"

- Wm Burroughs, *Naked Lunch*
- US Supreme Court 12/00

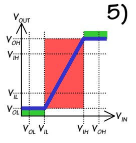
Handouts: Lecture Slides

The Need to be Discrete



Digital Information:

1) Quantize information into discrete units (bits)
2) Use voltages to represent information
3) To provide "noise margins" we had more strict requirements on device outputs than inputs
4) Nonlinear devices
5) Time behavior –

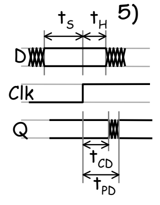


t_{CD} – upper bound on how long outputs remain valid after an invalid input

t_{PD} – lower bound on how long after inputs become valid before the output is guaranteed to be valid

Digital Time:

1) Need memory to compute values determined by current "state"
2) Quantize time into discrete events
3) Use clock edges to delineate (trigger) events
4) Provide "input stability" around edges
5) Time behavior –



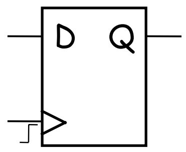
t_{SETUP} – lower bound on how long before a clock edge that the inputs must be stable

t_{HOLD} – lower bound on how long after a clock edge that the inputs must remain stable

Review of Dynamic Discipline

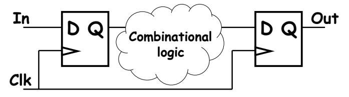
Synchronous 1-clock logic:

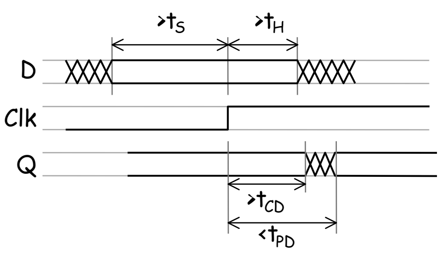
In order to build RELIABLE circuits we need only follow these simple rules.



An essential element of the DYNAMIC DISCIPLINE is the specification of setup and hold times relative to a global clock edge.

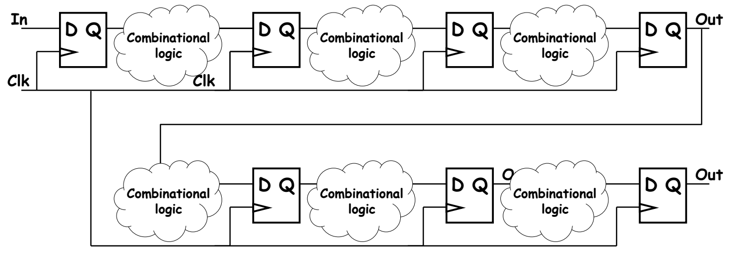
1) Clock period greater than: $FF(t_{pd})+Logic(t_{pd})+FF(t_{su})$
2) Satisfy hold time: $FF(t_{cd})+Logic(t_{cd})>FF(t_h)$
3) No combinational cycles





If we follow these simple rules…

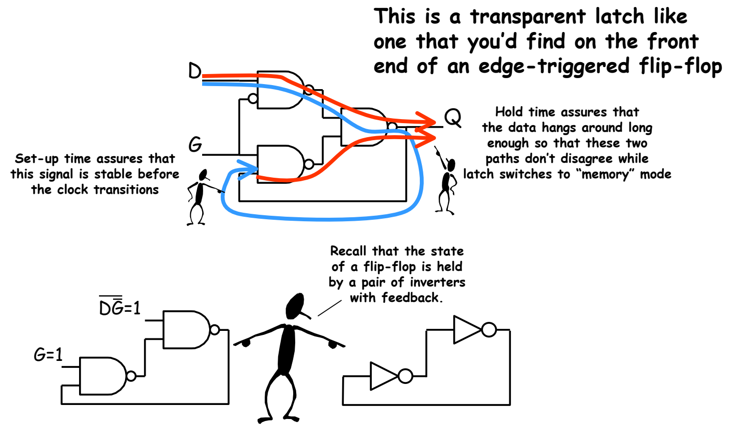
Can we guarantee that our system will always work?



With careful design we can make sure that the dynamic discipline is obeyed everywhere, except for the **Inputs**.

The World Doesn't Run on our Clock!

So, what happens if we miss an occasional set-up or hold time?

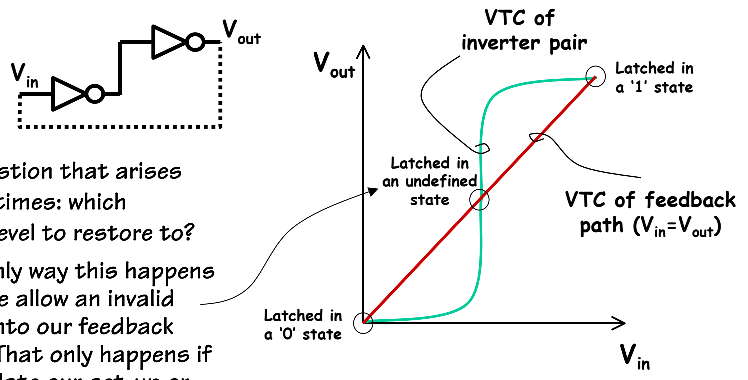

This is a transparent latch like one that you'd find on the front end of an edge-triggered flip-flop
D
G
Q
Set-up time assures that this signal is stable before the clock transitions
Hold time assures that the data hangs around long enough so that these two paths don't disagree while latch switches to "memory" mode
Recall that the state of a flip-flop is held by a pair of inverters with feedback.
$\overline{D}\overline{G}$=1
G=1

Persistent invalid states

One of the jobs of a digital gate is to restore questionable input signals to a valid output levels.

A question that arises sometimes: which valid level to restore to?

The only way this happens is if we allow an invalid level into our feedback loop. That only happens if we violate our set-up or hold times.



Metastability

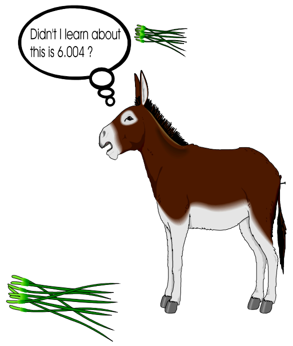
Metastability is the occurrence of a persistent invalid output. An unstable equilibria.

The idea of Metastability is not new:

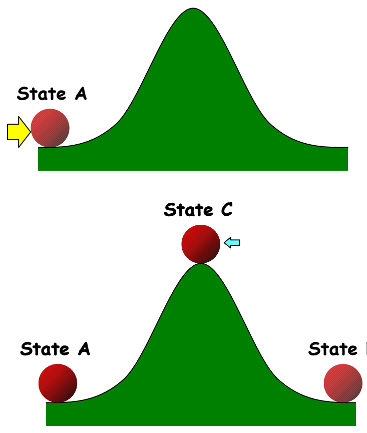
The Paradox of Buridan's Ass

Buridan, Jean (1300-58), French Scholastic philosopher, who held a theory of determinism, contending that the will must choose the greater good. Born in Bethune, he was educated at the University of Paris, where he studied with the English Scholastic philosopher William of Ockham (whom you might recall from his razor business). After his studies were completed, he was appointed professor of philosophy, and later rector, at the same university. Buridan is traditionally, but probably incorrectly, associated with a philosophical dilemma of moral choice called "Buridan's ass."

In the problem an ass starves to death between two alluring bundles of hay because it does not have the will to decide which one to eat.



Real-World Metastability



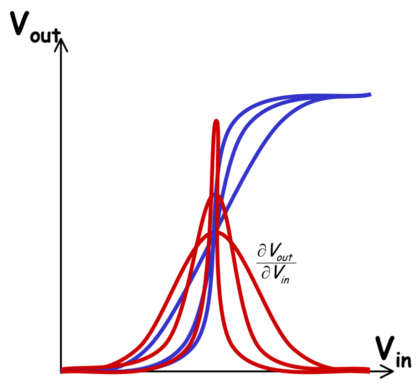
If we launch a ball up a hill we expect one of 3 possible outcomes:

a) Goes over
b) Rolls back
c) Stalls at the apex

That last outcome is not very stable.

- a gust of wind
- Brownian motion
- it doesn't take much

How do balls relate to digital logic?



Our hill is simply the derivative of the VTC (Voltage Transfer Curve).

Notice that the higher the gain thru the transition region, the steeper the peak of the hill. Thus, making it harder to get into a metastable state.

Nonetheless, Metastability will happen!

There's no easy solution

… so, embrace the confusion.

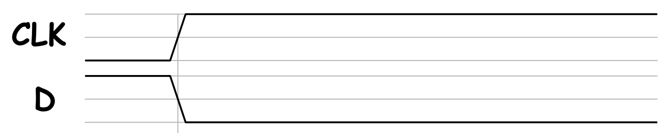
"Metastable States":

- Inescapable consequence of bistable systems
- Eventually a metastable state will resolve itself to valid binary level.
- However, the recovery time is UNBOUNDED … but influenced by parameters (gain, noise, etc)
- Probability of a metastable state falls off EXPONENTIALLY with time -- modest delay after state change can make it very unlikely.

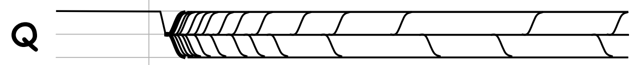
Our STRATEGY; since we can't eliminate metastability, we will do the best we can to keep it from contaminating the our designs

Observed Behavior:
typical metastable symptoms

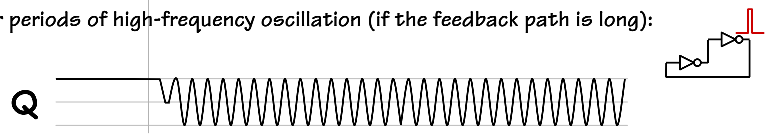
Following a clock edge on an asynchronous input:



We may see exponentially-distributed metastable intervals:



Or periods of high-frequency oscillation (if the feedback path is long):



Probability of system failure

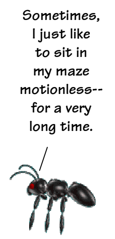
Product of the probabilities that

$$P_{failure} = P_M * P(\text{metastable state lasts} > t_w)$$

1) We get into a metastable state in the first place, P_M.

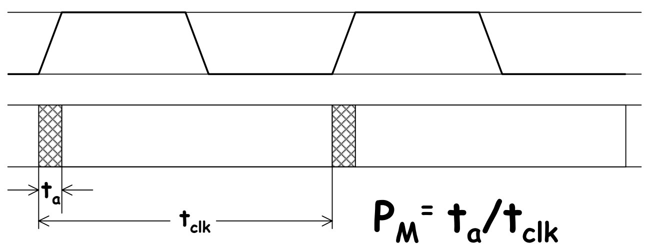
2) It persists long enough to corrupt other circuits, $P(\text{metastable state lasts} > t_w)$

Let's try to understand each of these terms.


Sometimes, I just like to sit in my maze motionless-- for a very long time.

Probability of Going Metastable

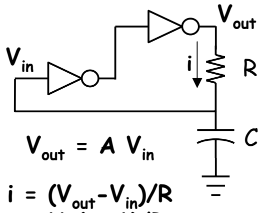
Basically, your input has to change within a critical window situated around the clock. This window is sometimes called the flip-flop's sampling *aperture*.



$$P_M = t_a/t_{clk}$$

A Model for Metastability

When metastable, the inverter's transistors operate in their *linear* regions (yep, that means 6.002).



We can model them as an amplifier with gain A. Furthermore, we will also need to model the resistance and capacitance of the feedback path.

$$V_{out} = A\ V_{in}$$

$$i = (V_{out} - V_{in})/R = V_{in}(A-1)/R$$

Then we can compute the transient response of the circuit as it leaves the metastable state.

$$C\ dV_{in}/dt = V_{in}(A-1)/R$$

$$V_{in} = V_{in}(0)e^{t(A-1)/RC}$$

$$V_{out} = AV_{in}(0)e^{t(A-1)/RC}$$

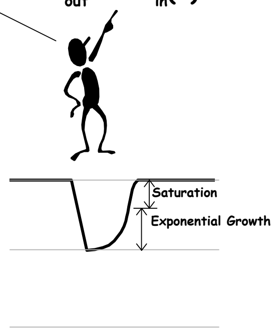
Probability of staying metastable

$$V_{out} = AV_{in}(0)e^{t(A-1)/RC}$$

$$\text{Let } \tau = RC/(A-1)$$

$$V_{out} = AV_{in}(0)e^{t/\tau}$$

This is the *initial* rate that the output will change. Eventually, the FETs saturate, and the curve slows down. But, once the process starts there's no stopping it.



We can find the initial voltage that allows the output to achieve a given level, V_{SAT}, within a prescribed period, t_w.

$$V_0 = V_{SAT}/(A\ e^{-tw/\tau})$$

$$P(V_{in}(0) < V_0) = V_0/V_{SAT} = 1/(A\ e^{-tw/\tau})$$

Now we know the probability that we will stay in a metastable state.

$$\boxed{P_{failure} = P_M/(A\ e^{-tw/\tau})}$$

Let's do some real numbers

$$\boxed{t_w = \tau \ln\left(\frac{P_M}{AP_{failure}}\right)}$$

Assume a 100 MHz clock that spends 10% of its time in the critical transition. (P_M=0.1)

Suppose our two inverters have a composite gain of 11 thru the invalid region of our VTC, and our feedback path has an RC constant of 10^{-9} (1 nS).

This yields a time constant $\tau = 10^{-10}$.

So, how long must we wait if we can tolerate 1 failure a year?

1 year	= 365 days
	= 8760 hours
	= 31536000 secs
	= 31536000 * 10^8 clks
$P_{failure}$	= 1/(31536000 * 10^8)
	= 3.171 * 10^{-16}

$$t_w = 10^{-10} \ln\left(\frac{0.1}{11*3.171*10^{-16}}\right)$$

$$t_w = 3.09 \text{ nS}$$

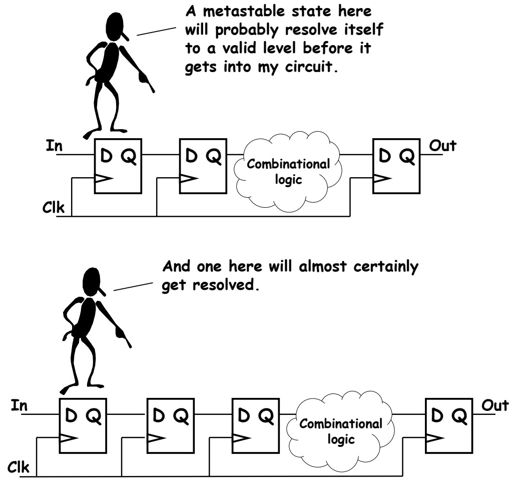
For 1 failure in 10 years

$$t_w = 3.33 \text{ nS}$$

Avoiding metastability

Synchronizers, extra flip flops between the asynchronous input and your logic, are the best insurance against metastable states.

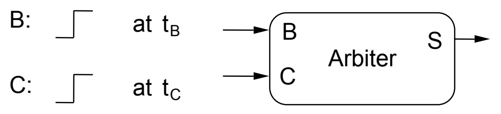
The higher the clock rate, the more synchronizers should be considered.


A metastable state here will probably resolve itself to a valid level before it gets into my circuit.
In
D Q
D Q
Combinational logic
D Q
Out
Clk
And one here will almost certainly get resolved.
In
D Q
D Q
D Q
Combinational logic
D Q
Out
Clk

The Asynchronous Arbiter:

a classic problem

UNSOLVABLE

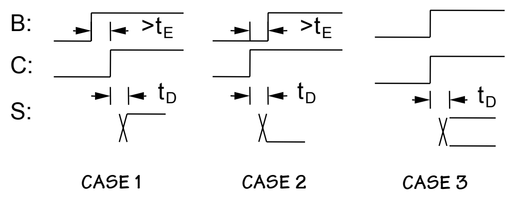


For NO finite value of t_E and t_D is this spec realizable, even with reliable components!

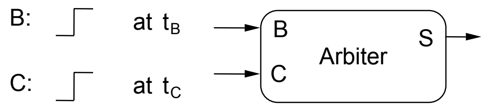
Arbiter specifications:

- finite t_D (decision time)
- finite t_E (allowable error)
- value of S at time t_C+t_D:

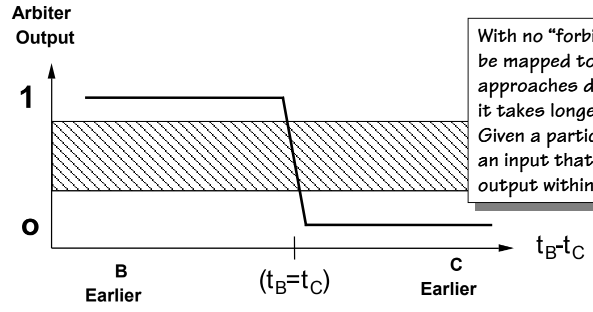
1	if $t_B < t_C - t_E$
0	if $t_B > t_C + t_E$
0, 1	otherwise



Why? Metastability (again)!



Issue: Mapping the *continuous* variable $(t_B - t_C)$ onto the *discrete* variable S in bounded time.

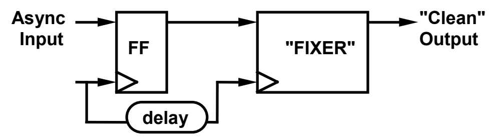


With no "forbidden zone," all inputs have to be mapped to a valid output. As the input approaches discontinuities in the mapping, it takes longer to determine the answer. Given a particular time bound, you can find an input that won't be mapped to a valid output within the allotted time.

The Perfect Synchronizer:

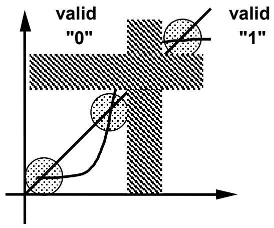
our own "perpetual motion" machine

Bad Idea # 1: Detect metastable state & Fix



Bug: detecting metastability is itself subject to metastable states, i.e., the "fixer" will fail to resolve the problem in bounded time.

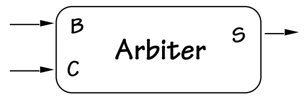
Bad Idea #2: Define the problem away by making metastable point a valid output



Bug: the memory element will flip some valid "0" inputs to "1" after a while.

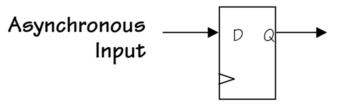
What we CAN'T build

1. Bounded-time Asynchronous Arbiter:



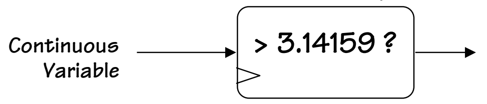
S=0 iff B edge first, 1 iff C edge first,
1 or 0 if nearly coincident
S valid after t_{pd} following (either) edge

2. Bounded-time Synchronizer:



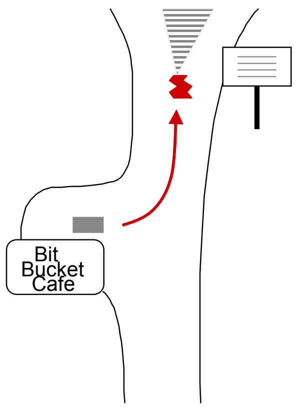
Output = D at active clock edge, either 1 or 0
iff D invalid near clock edge
Q valid after t_{pd} following active clock edge

3. Bounded-time Analog Comparator:



0 or 1,
finite t_{pd}

Every-day Metastability - I



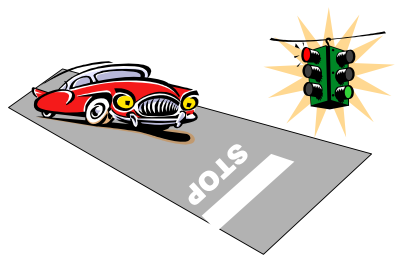
Ben tries the famous "6.004 defense":

Ben leaves the Bit Bucket Café and approaches fork in the road. He hits the barrier in the middle of the fork, later explaining "I can't be expected to decide which fork to take in bounded time!".

Is the accident Ben's fault?

Every-day Metastability - II

GIVEN:
- Normal traffic light: GREEN, YELLOW, RED sequence
- 55 MPH Speed Limit
- Sufficiently long YELLOW, GREEN periods



IS IT POSSIBLE TO RELIABLY AVOID RUNNING A RED LIGHT ...
1. Under ANY circumstances
2. Without blocking intersection
3. Assuming you don't slow down when the light is continuously GREEN

PLAUSIBLE STRATEGIES:

A. Move at 55. At calculated distance D from light, sample color (using an unbounded-time synchronizer). GO ONLY WHEN stable GREEN.

B. GO on GREEN; STOP otherwise.

C. Stop 1 foot before intersection. On GREEN, gun it.

DOES ANY STRATEGY WORK???

Summary

Metastability will happen… don't ignore it, deal with it.

As a system designer…

1) Delay after sampling asynchronous inputs (sychronizers)
2) Use "slow" clocks with "fast" rise times (reduce P_M)
3) If possible, predict close timing races and decide them in advance (eg, if you have two periodic clocks)
4) Avoid the problem altogether:
   - Use single clock, obey dynamic discipline
   - Avoid state. Combinational logic has no metastable states!
   - Build entirely asynchronous circuits

As a circuit designer…

1) Maximize gain (restoration rate) of synchronizers
2) Optimize feedback paths in synchronizers