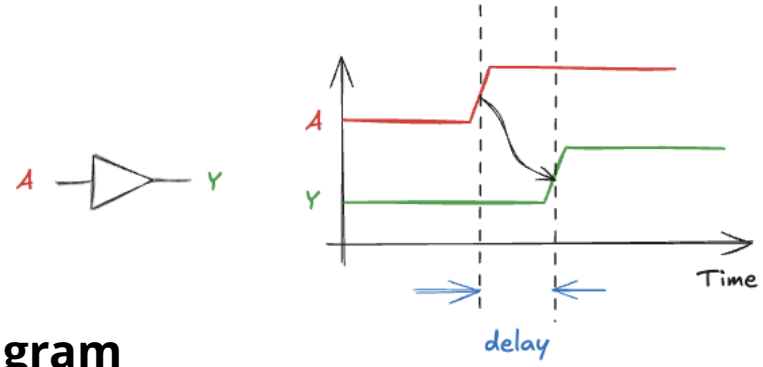
Timing

Index

- Combinational Circuits Timing
  - Propagation and Contamination Delay
  - Critical and short paths
  - Hazards and Glitches
  - Wire delay
- Timing of sequential logic
  - Dynamic Discipline
  - Aperture Time
  - Clock-to-Q Delays
- System Timing
  - Setup Time Constraint
  - Hold Time Constraint
  - Timing Analysis
  - Fixing Hold Time Violations
  - Clock Skew
- Timing violations
  - Metastability
  - Resolution Time
  - Synchronizers
  - Reliability

Delay

- One of the most challenging issues in circuit design is timing
  - **making a circuit run fast**
- An output **takes time** to change in response to an input change
  - circuit delays are on the order of **picoseconds** (10^{-12}) to nanoseconds (10^{-9})

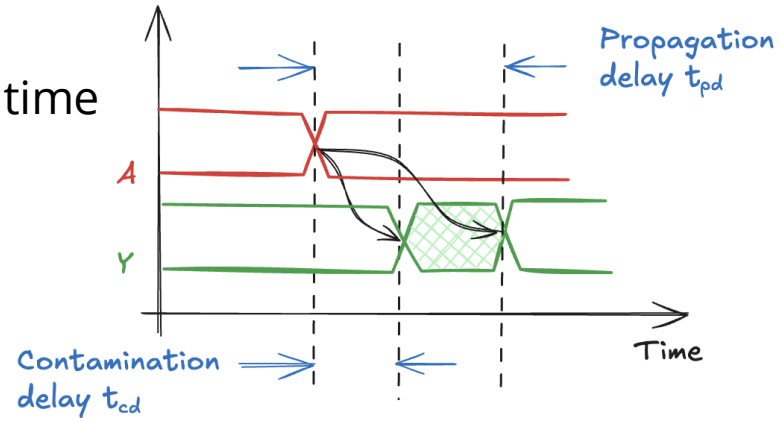


- **Timing diagram**
  - the **transient response** of the buffer circuit when an input changes
  - transition from LOW to HIGH is called the **rising edge**
  - transition from HIGH to LOW is called the **falling edge**
  - The arrow indicates that the rising edge of Y depends on the rising edge of A
- **Delay** is measured from 50% point of the input to 50% point of the output

Propagation and Contamination Delays

- **propagation delay** t_{pd}
  - **maximum** time from when the input changes until the output reaches its final value
- **contamination delay** t_{cd}
  - **minimum** time from when the input changes until the output starts to change its value

- A is initially either HIGH or LOW
  and changes to the other state at a particular time
  - we are interested only in the fact that
    it changes, not what value it has
- In response, Y changes later
  - Y **starts** to change t_{cd} after A transitions
  - Y **definitely** settles to its new value within t_{pd}



- The causes of delay include the **time required to charge the capacitance** in a circuit and **the speed of light**
  - calculating t_{pd} and t_{cd} requires delving into the lower levels of abstraction
  - manufacturers supply **data sheets** specifying these delays for each gate

Critical path

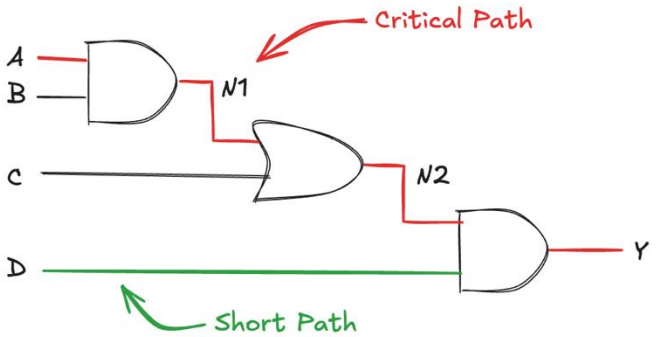
- Delays are also determined by the **path** a signal takes from input to output

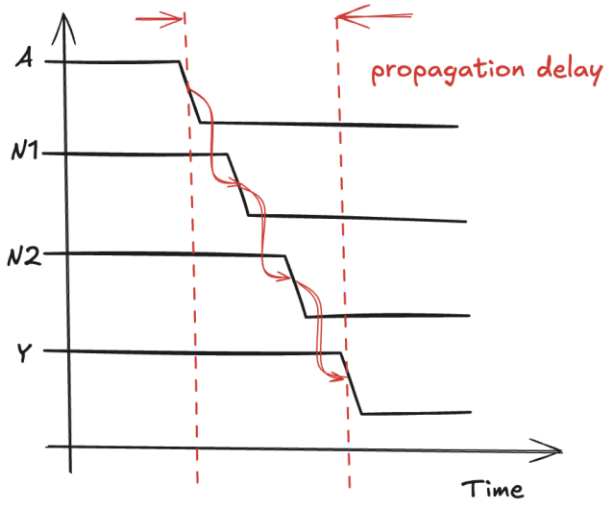
- The **critical path** is the **longest** (**slowest**) path
  - the input travels through more gates to the output
  - it **limits the speed** at which the circuit operates
  - the propagation delay of a circuit is the **sum of the propagation delay of all element on the critical path**

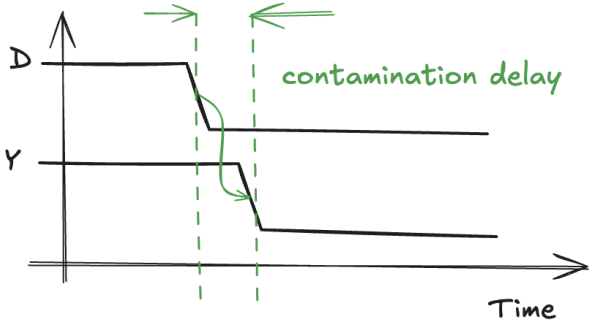
$$t_{pd} = t_{pd,AND} + t_{pd,OR} + t_{pd,AND}$$

- The **short path** is the **shortest** (**fastest**) path
  - the input travels through the minimum number of gates to the output
  - the contamination delay is the **sum of the contamination delays**

$$t_{cd} = t_{cd,AND}$$

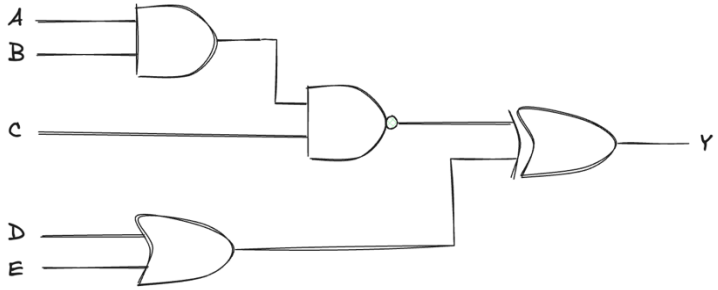






Finding delays

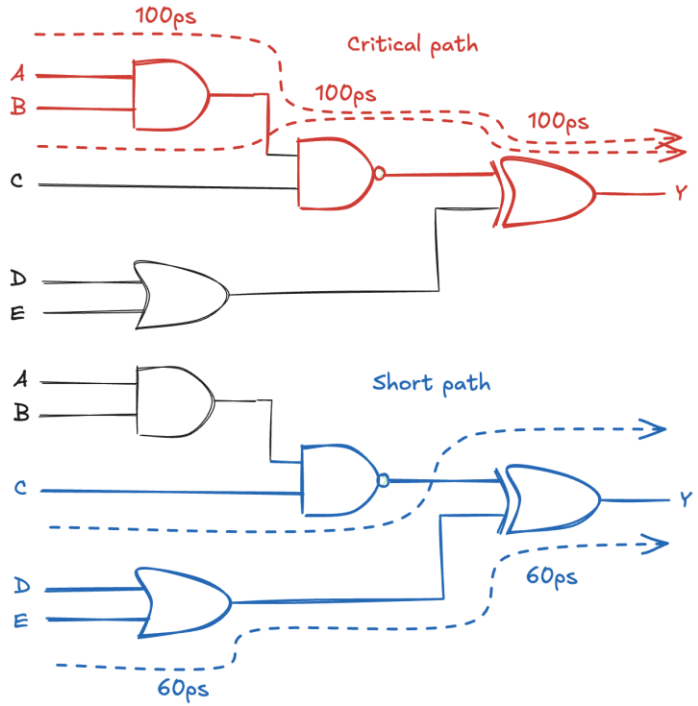
- Find the propagation delay and contamination delay of the following circuit



- According to the datasheets, each gate has a **propagation delay of 100ps** and a **contamination delay of 60ps**

- The critical path is from input A or B through three gates to the output Y
  - hence, t_{pd} is three times the propagation delay of a single gate, 300ps

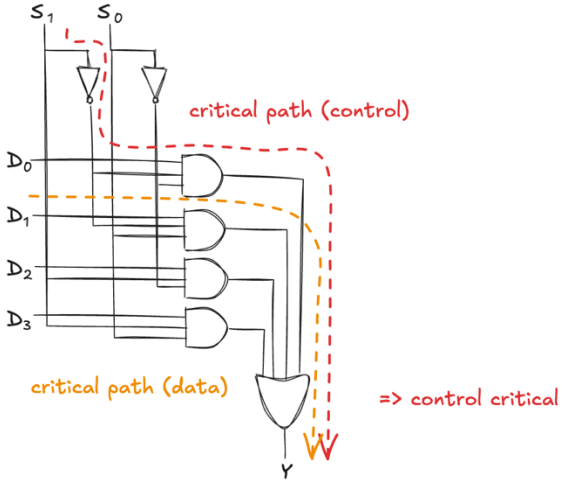
- The shortest path is from input C, D, or E through two gates to the output Y
  - so t_{cd} is 120ps

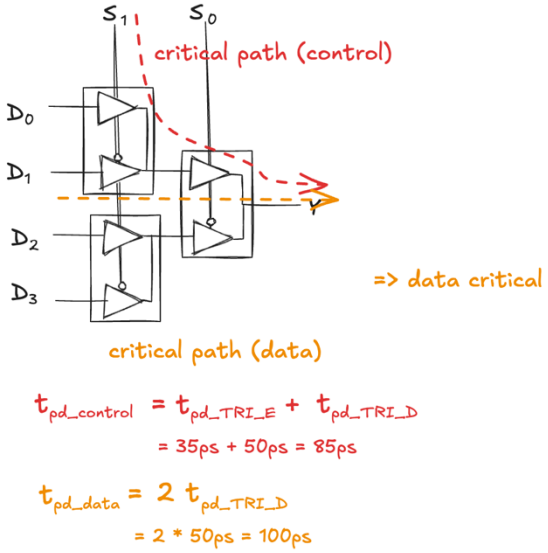


Control-critical vs data-critical

- Critical paths can be:
  - **control critical:** the critical path is from the control to the output
    - if data arrive before the control, we prefer the design with the shortest control delay
  - **data critical:** the critical path is from the inputs to the output
    - if the control arrive before the data, we prefer the design with the shortest data delay
- Compare two four-input multiplexer designs (two-level logic vs multiplexer logic)

Gate	tpd (ps)
NOT	30
2-input AND	60
3-input AND	80
4-input OR	90
Tristate (A to Y)	50
Tristate (Enable to Y)	35

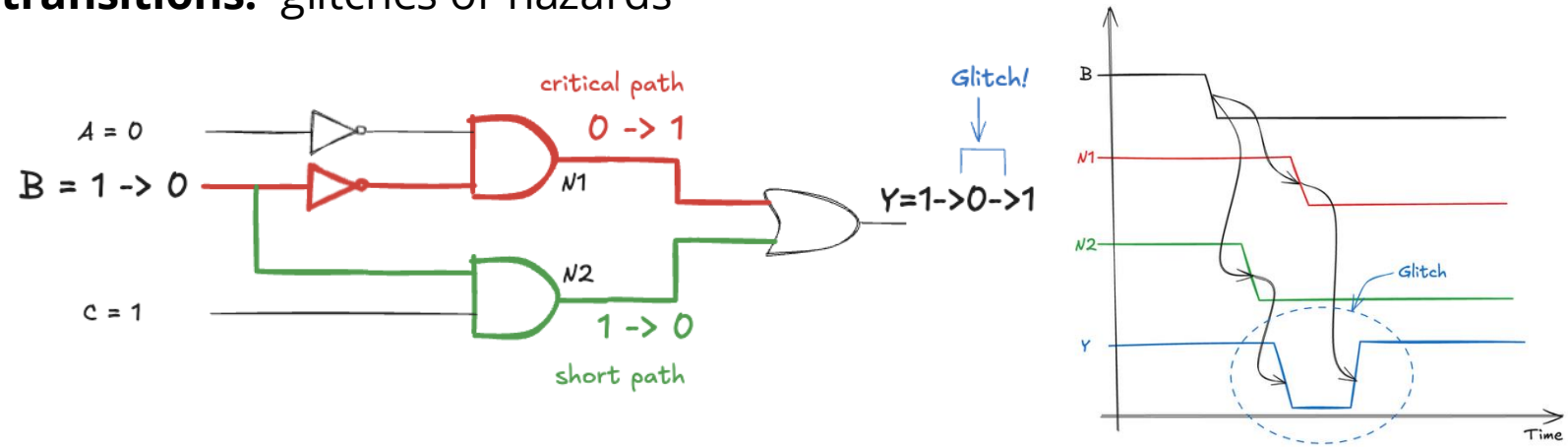




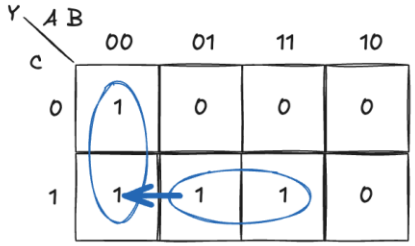
- The best choice depends on the power, cost, and availability of parts

Glitches (1)

- It is possible that a **single input transition** can cause **multiple output transitions:** glitches or hazards



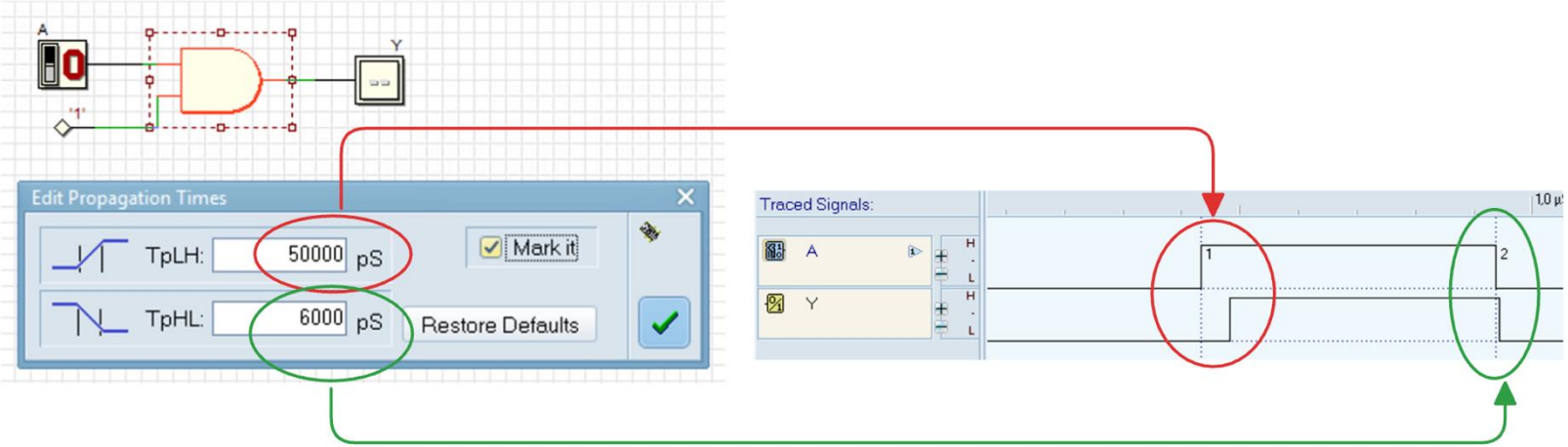
- Transition across the boundary of two prime implicants indicates a possible glitch



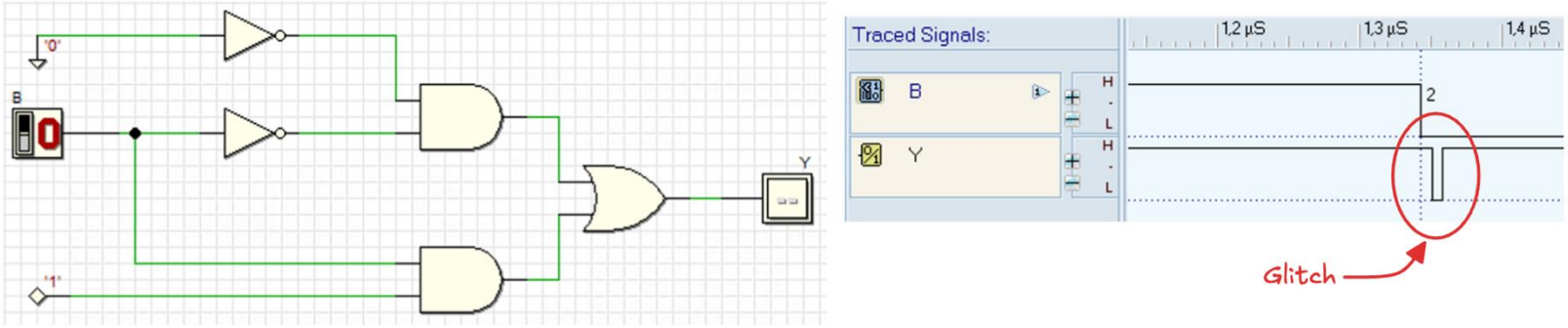
- If we **wait** for the propagation delay to elapse **before we depend on the output**, glitches **are not a problem**

Glitches (2)

- The DEEDS simulator consider the propagation delay (5ns):

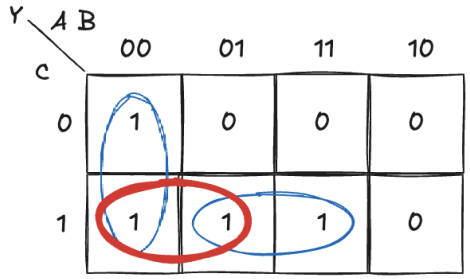


- and this glitch effect can be seen in the simulation (zooming enough):

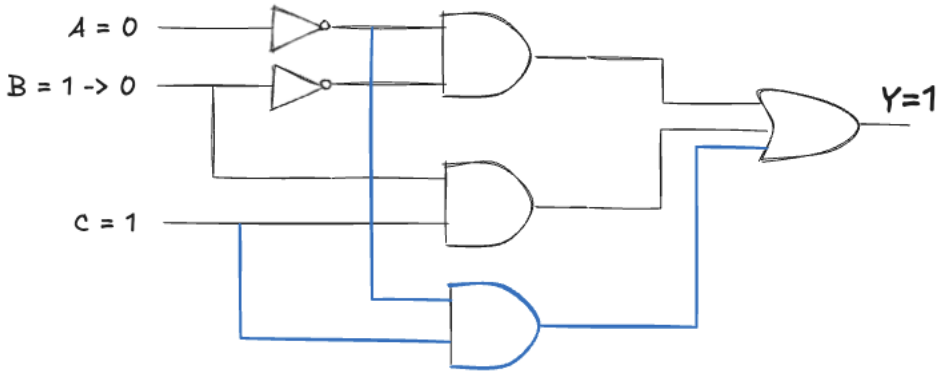


Glitches (3)

- We can eliminate glitches by adding **redundant implicants** to the K-map to cover these boundaries
  - at the cost of extra hardware



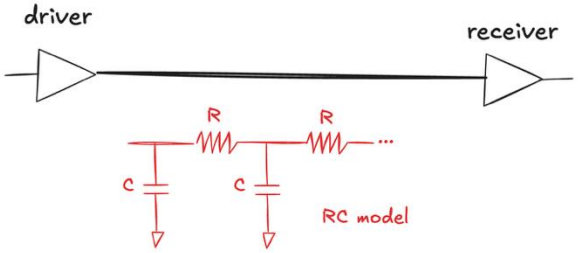
$Y = \bar{A}\bar{B} + Bc + \bar{A}c$



- However, **simultaneous transitions on multiple inputs** can also cause glitches that cannot be fixed by adding hardware
- Most systems have these simultaneous transitions, then **glitches are a fact of life** in most circuits
- The point is **not to eliminate them** but to **be aware that they exist**
  - this is especially important when looking at timing diagrams on a simulator or oscilloscope

Wire delay

- We have assumed that **wires are equipotential connections that have a single voltage along their entire length**
- However, signals **propagate** at the speed of light (electromagnetic waves)
  - if wires are short enough or signals change slowly, the assumption is good enough
  - if the wire delay is shorter than a fraction (typically 20%) of the edge rates (rise or fall times) of a signal it has an **insignificant effect**
- When **wire length increases**, causing significant resistance and capacitance, we can model the wire with an **RC model:**



- When wires are long or **signals are very fast**, we should consider **wave propagation effects** modelled as **transmission lines**
  - in which a **wave** of voltage and current propagates at the speed of light
  - when the wave reaches the end of the line, it may **reflect** along the line, causing noise and odd behaviours

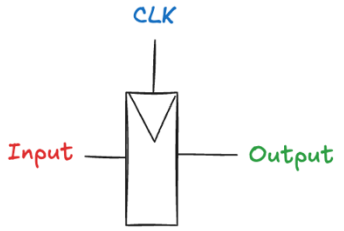
Timing of sequential logic

- A flip-flop copies input D to output Q on the rising edge of the clock (**sampling**)
  - if D is stable (0 or 1) when the clock rises, the behaviour is clearly defined
  - what happens if D is changing at the same time the clock rises?
- The problem is like that faced by a camera when snapping a picture
  - it is characterized by its aperture time, during which the object must remain still for a sharp image to be captured
- A sequential element has an **aperture time** around the clock edge, during which the input **must be stable** for the flip-flop to produce a well-defined output
  - **setup time** before the clock edge
  - **hold time** after the clock edge
- As the **static discipline** (limits us to using logic levels outside the forbidden zone), the **dynamic discipline** limits us to signals that **change outside the aperture time**
  - we can think of time in **discrete units** called **clock cycles**, just as we think of signal levels as discrete 1's and 0's
  - a signal may glitch and oscillate for some bounded amount of time, we are concerned only about its final value at the end of the clock cycle

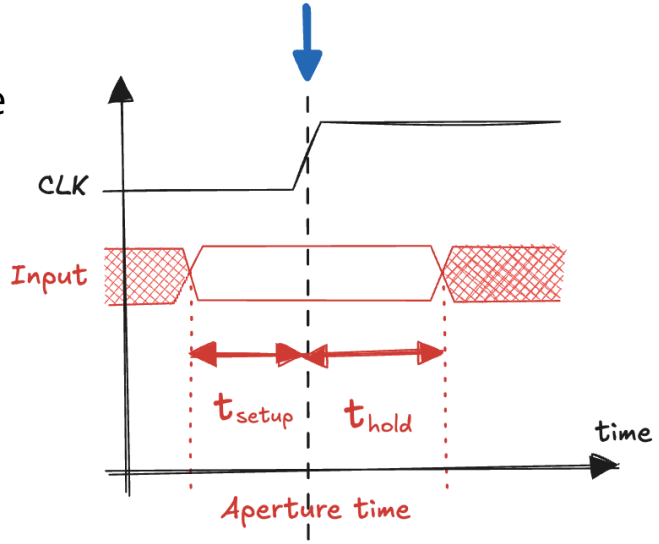
Dynamic Discipline

- We can write **A[n]** as the value of signal A at the end of the n-th clock cycle, rather than A(t), where t is any real number
  - the clock period must be long enough for all signals to settle
  - a **limit on the speed of the system**
- In real systems, the clock does not reach all flip-flops at precisely the same time
  - **clock skew**
  - further increases the necessary clock period
- Sometimes it is **impossible to satisfy the dynamic discipline**, especially when interfacing with the real world
  - consider a circuit with an input coming from a button, we might press the button just as the clock rises
  - this result in a phenomenon called **metastability**
    - flip-flop captures a value between 0 and 1
    - that can take an unlimited amount of time to resolve into a good logic value

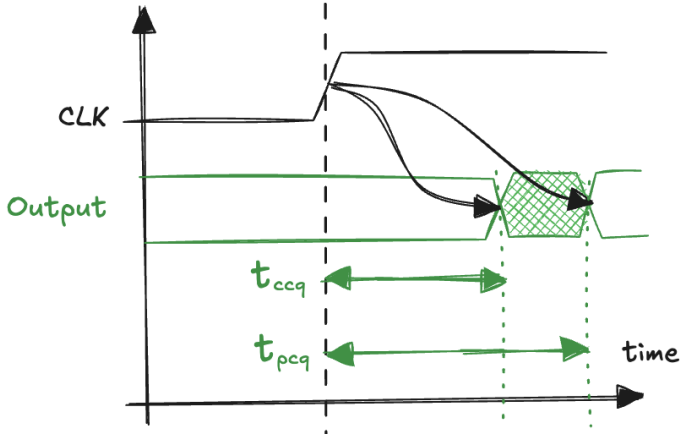
Timing Specification



- The **input**
  - must have **stabilized** at least **setup time** before the rising edge of the clock
  - and it must **remain stable** for at some **hold time** after the rising edge of the clock
  - the sum of setup and hold times is called **aperture time**, because it is the total time for which the input must remain stable

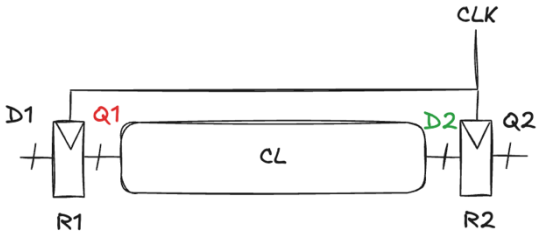


- The **output**
  - will **start to change** after the **clock-to-Q contamination delay**
    - the fastest delays through the circuit
  - and it will **settle to the final value** within the **clock-to-Q propagation delay**
    - the slowest delays through the circuit

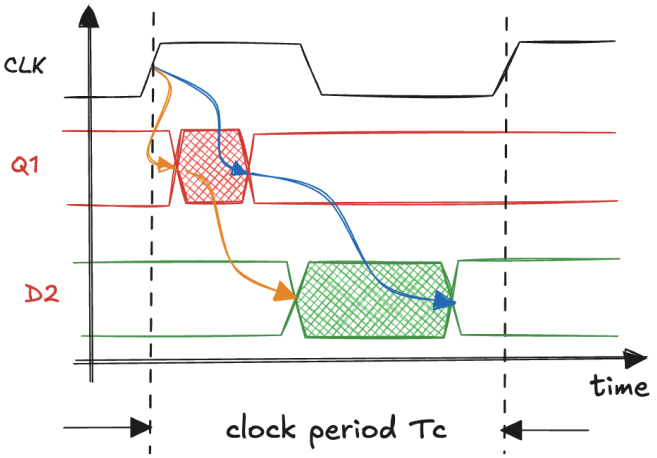


System Timing

- The **clock period** (Tc) is the time between rising edges of a repetitive clock signal
- Its reciprocal fc = 1/Tc is the **clock frequency**
  - measured in units of **Hertz** (Hz) or **cycles per second**
- Decreasing Tc (increasing fc) increases the work a digital system can accomplish per unit time
- Consider a generic path in a synchronous sequential circuit

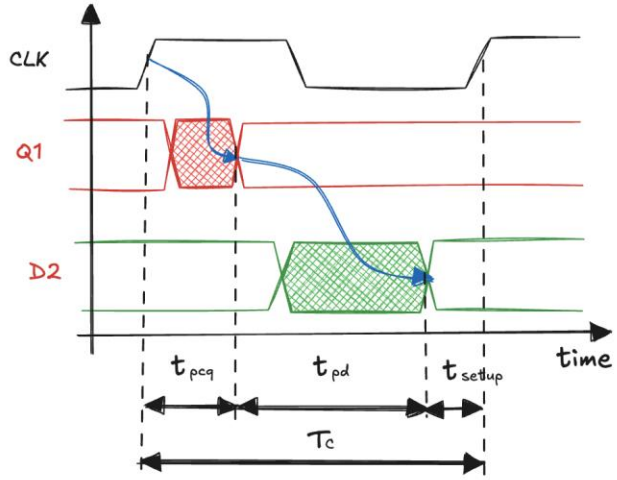


- Output signals may start to change a contamination delay after input changes
  - orange arrows
- Output signals settles to the final value within a propagation delay after its input settles
  - blue arrows



Setup Time Constraint

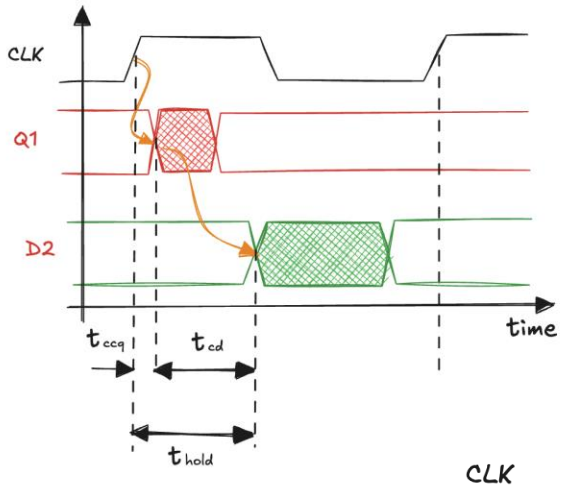
- We can analyse the timing constraints with **respect to the setup time** of the second register
  - we need to wait the propagation time of R1
  - we need to wait the propagation time of the combinational logic
  - to satisfy the requirements of R2, D2 must settle no later than setup time before the next clock edge
  - minimum clock period: $Tc > t_{pcq} + t_{pd} + t_{setup}$



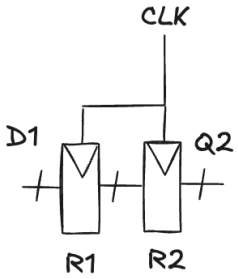
- In commercial designs, usually the only variable under the control of the designer is the **maximum propagation delay through the combinational logic**
  - clock period is dictated by the marketing department (to ensure a competitive product)
  - the flip-flop delay and setup time are specified by the manufacturer
  - $\mathbf{t_{pd} < Tc - (t_{pcq} + t_{setup})}$
- Ideally, the entire cycle time Tc would be available for useful computation in the combinational logic. The **sequencing overhead** ($t_{pcq} + t_{setup}$) cuts this time

Hold Time Constraint

- We can analyse the timing constraints with **respect to the hold time** of the second register
  - to satisfy the requirements of R2, D2 must not change until hold time after the rising edge of the clock
  - $t_{ccq} + t_{cd} > t_{hold}$
  - **minimum contamination delay through the combinational logic: $t_{cd} > t_{hold} - t_{ccq}$**



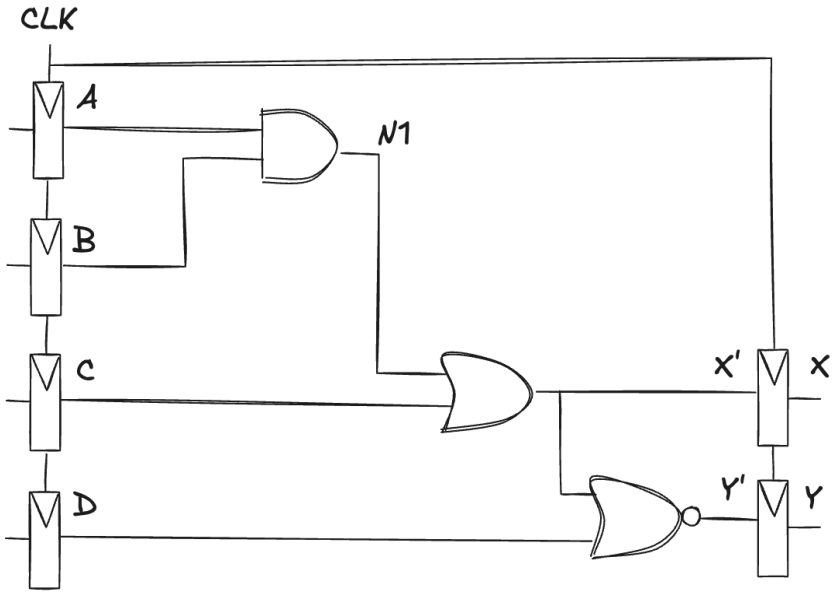
- We expect that two flip-flops may be directly cascaded
  - t_{cd}=0 (no combinational logic) and $t_{hold} < t_{ccq}$
  - reliable flip-flop must have hold time shorter than contamination delay
  - often flip-flops are designed with **t_{hold}=0**



- We usually **ignore the hold time constraint**
  - nevertheless, it is **critically important**
  - if violated, the only solution is to increase the contamination delay through the logic
    - which requires redesigning the circuit
  - unlike setup time constraint, it cannot be fixed by adjusting the clock period

Timing Analysis (1)

- Sequential circuits have setup and hold time constraints that dictate the **maximum** and **minimum** delays of the combinational logic between flipflops
  - the maximum delay constraint limits the number of consecutive gates on the critical path of a circuit
- Consider the circuit in figure and determine the maximum clock frequency and whether any time violations could occur



Flip-flop:

t_{ccq} = 30ps

t_{pcq} = 80ps

t_{setup} = 50ps

t_{hold} = 60ps

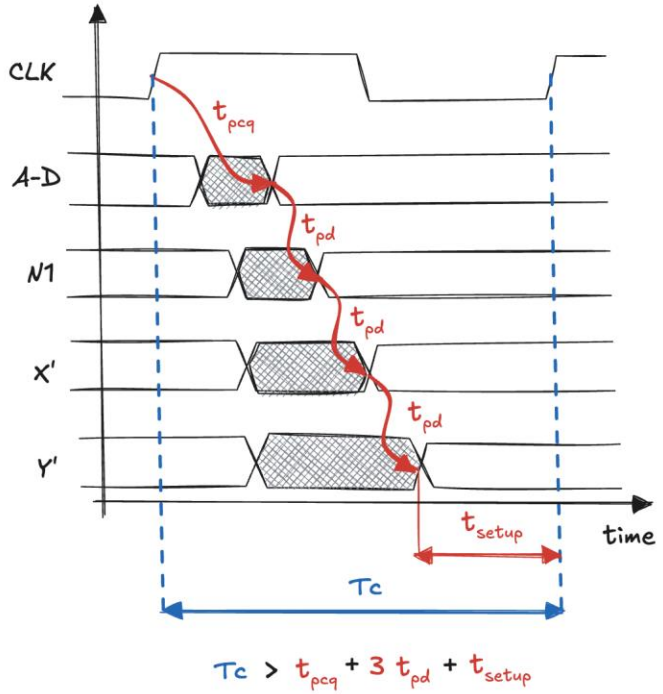
Logic gates:

t_{cd} = 25ps

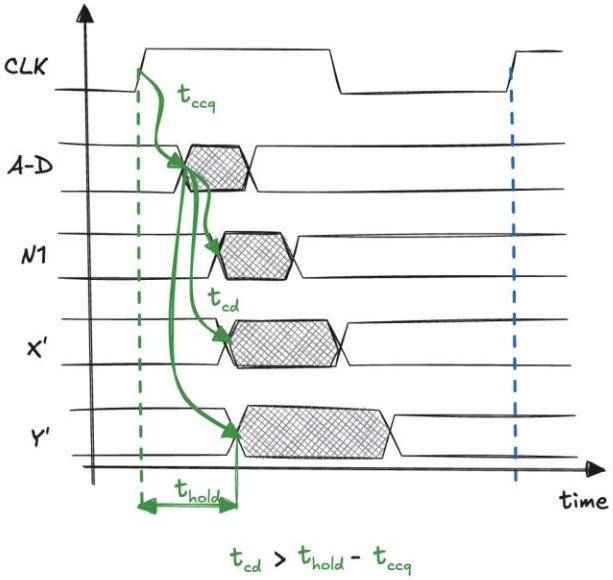
t_{pd} = 40ps

Timing Analysis (2)

- Critical path:
  - gates require same propagation delay
  - three gate delays: A/B -> n1 -> X’ -> Y’
  - Y' must set up before the next CLK rising edge
- Maximum clock frequency**:**
  - Tc > 80ps + 3*40ps + 50ps = 250ps
  - fc = 1/Tc = 4 GHz

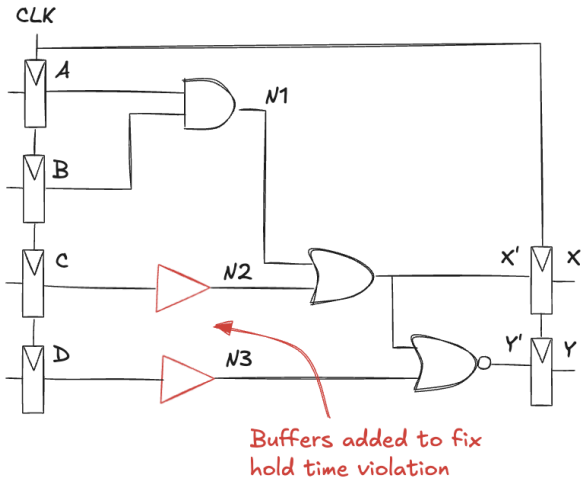


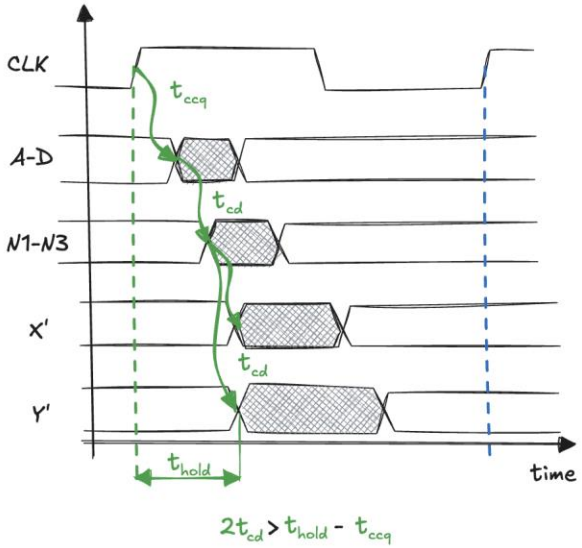
- Shortest path:
  - we need to consider contamination delay
  - only one gate delay: C -> X’ (D –> Y’)
- Hold Time Constraint
  - t_{cd} > 60ps – 30ps = 30ps (25 ps)
  - the circuit has a **hold time violation** and may behave erratically at any clock frequency



Fixing Hold Time Violations

- A possible solution is to **add buffers** to slow down the short paths

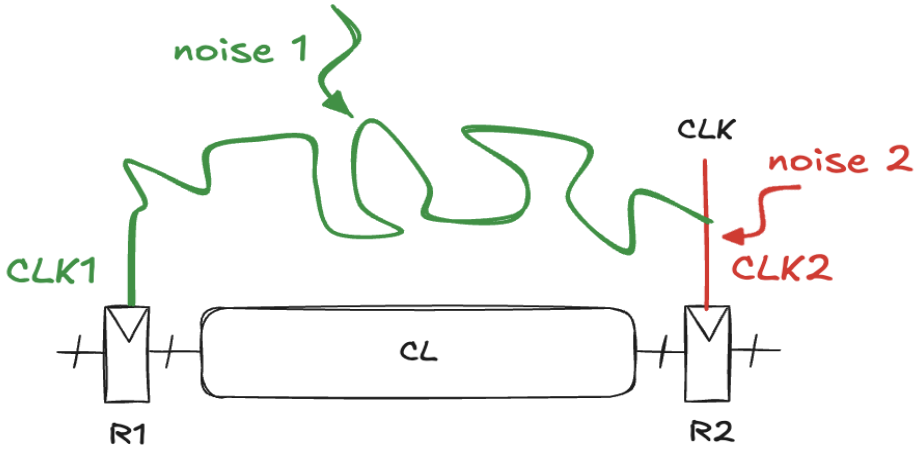


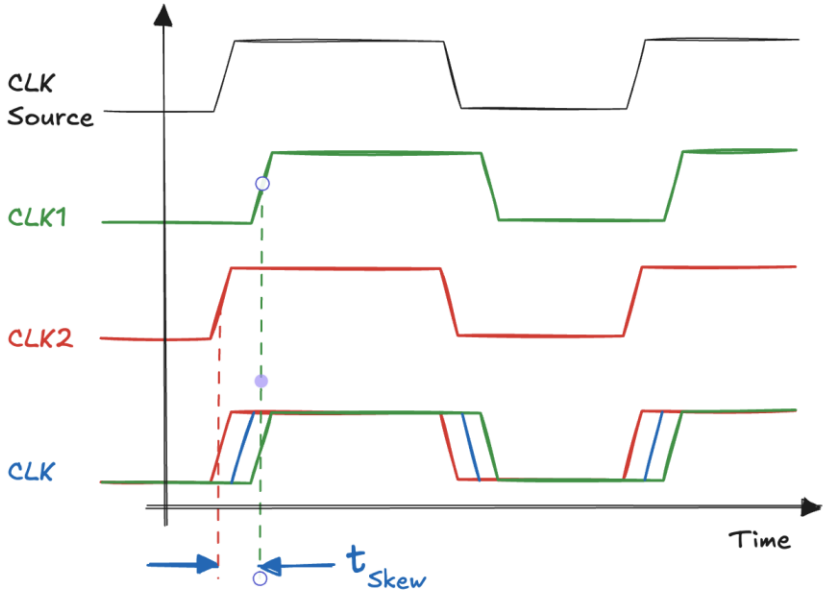


- Short paths are slowed by the buffer
  - $t_{ccq} + 2*t_{cd} = 30ps + 2*25ps = 80ps > t_{hold}$
- Critical path is unaffected (it not pass through buffers)
  - the maximum clock frequency is still 4 GHz
- In general, **adding buffers** can usually (not always) solve hold time problems without slowing the critical path
  - the example had an **unusually long hold time** to illustrate the problems
  - most flip-flops are designed with $t_{hold} < t_{ccq}$ to avoid such problems

Clock Skew (1)

- We assumed that the clock **reaches all registers at the same time**
- However, we can have **some variation** in this time
  - wires from clock source to different registers may be of different lengths
  - noise can be different on different paths

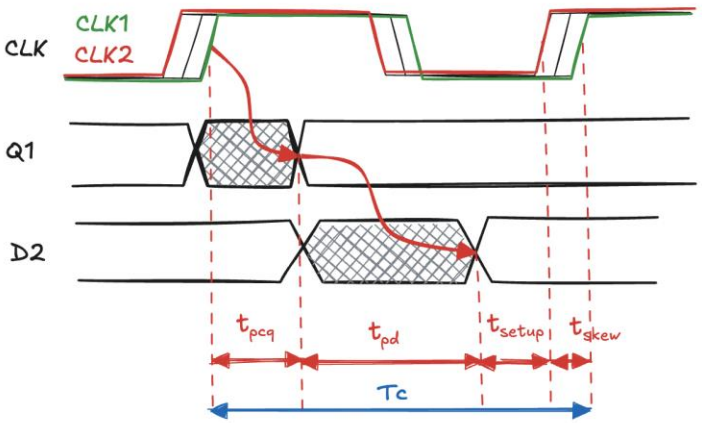




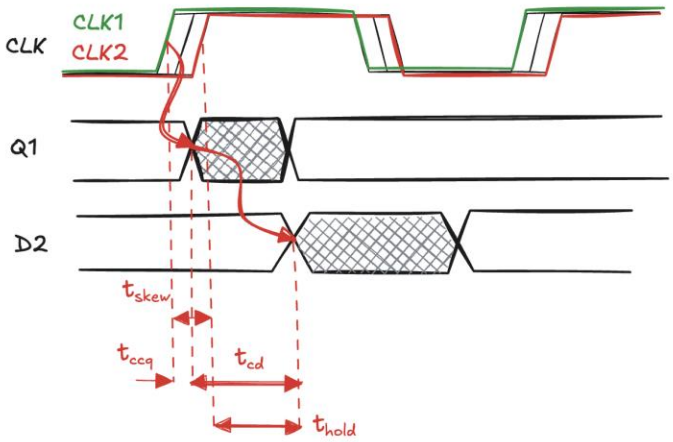
- When doing timing analysis, we must consider the **worst-case scenario** so that we can guarantee that the circuit will **work under all circumstances**

Clock Skew (2)

- Consider the **setup time constraint** worst case
  - R1 receives latest clock, R2 receives earliest clock
  - less time for data propagation
  - $Tc > t_{pcq} + t_{pd} + t_{setup} + t_{skew}$
  - $t_{pd} < Tc - (t_{pcq} + t_{setup} + t_{skew})$



- Consider the **hold time constraint** worst case
  - R1 receives early clock, R2 receives a late clock
  - D2 must not change until hold time after the late clock
  - $t_{ccq} + t_{cd} > t_{hold} + t_{skew}$
  - $t_{cd} > t_{hold} + t_{skew} - t_{ccq}$



- Clock skew **increases** setup time and the hold time
  - it reduce the time available for useful work in the combinational logic
  - increases the minimum delay through the combinational logic: even if $t_{hold} = 0$, a pair of back-to-back flip-flops will violate the constrain if $t_{skew} > t_{ccq}$

Clock Skew (3)

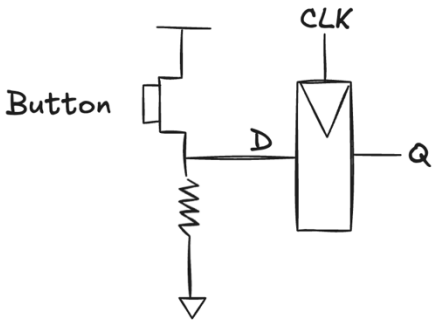
- Revisit the timing analysis example **assuming a 50ps of clock skew**
- Critical path remains the same
  - setup time is increased by the skew
  - $Tc > t_{pcq} + t_{pd} + t_{setup}$ **+ t_{skew}** = 80ps + 3*40ps + 50ps **+ 50ps** = 300ps
  - fc = 1/Tc = 3.33 GHz
- Short path also remains the same at 55ps
  - hold time is increased by the skew
  - $t_{hold} + t_{skew}$= 60 + 50 = 110 ps, which is much greater than 55 ps
  - **skew just makes the violation worse**
- To fix the problem:
  - more buffers could be inserted
  - buffers would need to be added on the critical path as well, reducing the clock frequency
  - alternatively, a flip-flop with a shorter hold time might be used

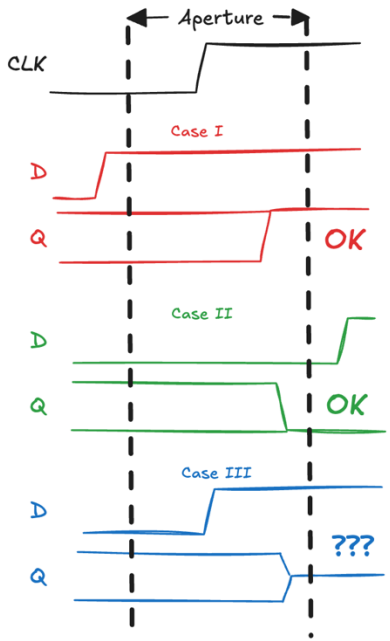
Metastability (1)

- It is not always possible to **guarantee** that the input to a sequential circuit is stable during the aperture time
  - especially when the input arrives **from the external world**

- Consider a button connected to the input of a flip-flop
  - when the button is **pressed much before** CLK, Q = 1
  - when the button is **not pressed until long after** CLK, Q = 0
  - when the button is **pressed between t_{setup} before CLK and t_{hold} after CLK**, the input **violates the dynamic discipline** and the output is undefined

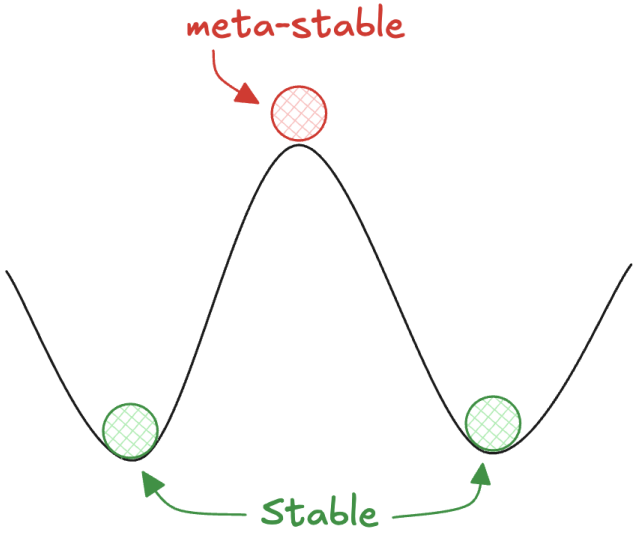
- When a flip-flop samples an input changing during the aperture, output may **momentarily** go in forbidden zone
  - **metastable state**
  - eventually, it will resolve to a stable state, however, the **resolution time** required is **unbounded**



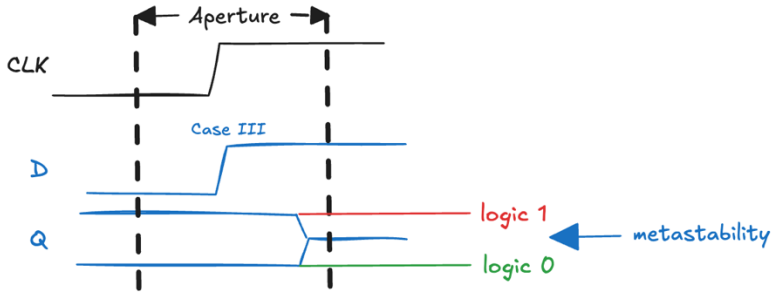


Metastability (2)

- The metastable state of a flip-flop is analogous to a ball on the summit of a hill between two valleys
  - valleys are stable states
    - the balls remain there
  - the top of the hill is metastable
    - the ball would remain there if it were perfectly balanced, but because nothing is perfect, the ball will eventually roll to one side or the other
    - the time required for this change to occur depends on how nearly well balanced the ball originally was



- Every bistable device has a metastable state between the two stable states
- Metastable value **propagates** to other parts of the system



Resolution Time

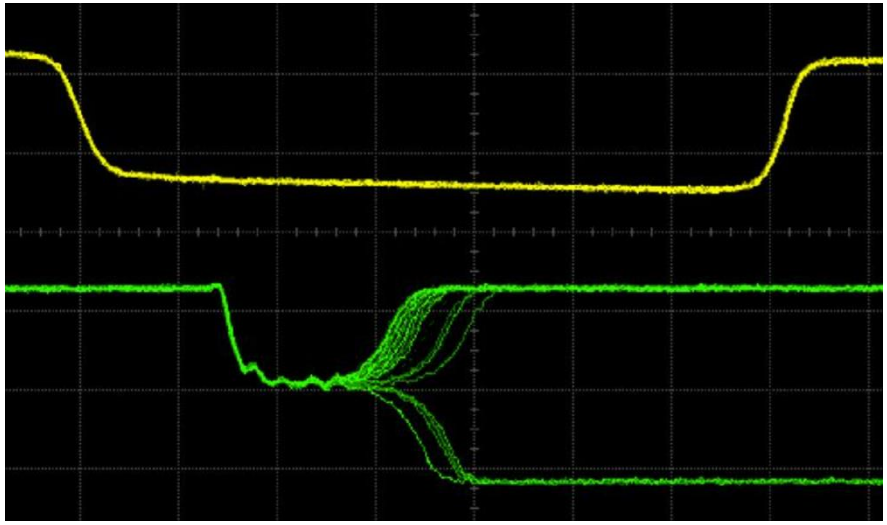
- The resolution time is a **random variable**
- Theoretical and experimental analyses have shown that the **probability** that the resolution time exceeds some arbitrary time t **decreases exponentially** with

$$\text{Prob}(t_{\text{res}} > t) = \frac{T_0}{T_C} \cdot \exp\left(-\frac{t}{\tau}\right)$$

  - where T_C is the clock period
  - T_0 and τ are characteristic of the flipflop
  - the equation is valid only for t substantially longer than t_{pcq}

- This means that time required to resolve is **unbounded**

- However, if **we wait long enough** the flip-flop will reach a valid logic level with high probability

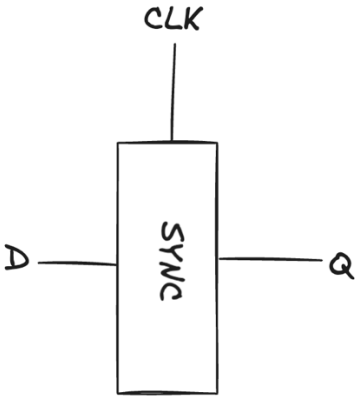


Synchronizers (1)

- **Asynchronous inputs** to digital systems from the real world are **inevitable**
  - human input is asynchronous
  - can lead to metastable voltages and erratic system failures

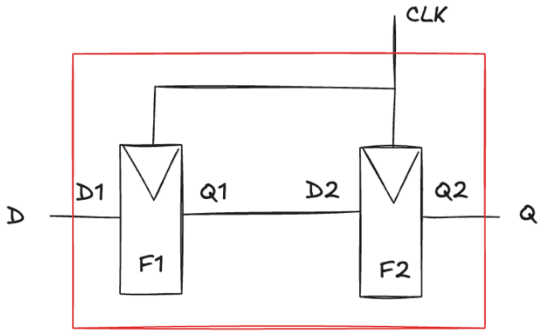
- **Ensure that the probability of encountering a metastable voltage is "small"**
  - cell phone: one failure in 10 years is acceptable, the user can turn the phone off and on
  - medical device, one failure in the expected life of the universe is a better target

- A **synchronizer** is a device that receives an asynchronous input D and a clock CLK and **produces a valid output Q with high probability**
  - If D is stable during the aperture, Q should take on the same value as D
  - If D changes during the aperture, Q may take HIGH or LOW, but must not be metastable

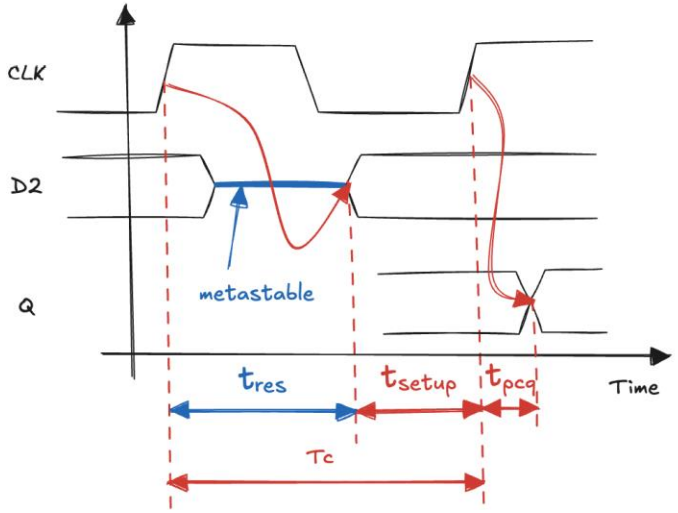


Synchronizers (2)

- We can build a simple synchronizer out of two flip-flops
  - F1 samples D on the rising edge of CLK
  - if D is changing at that time, the output Q1 (and input D2) may be momentarily metastable
  - if **clock period is long enough**, D2 (with high probability), resolve to a valid logic level
  - so, F2 samples D2 (now stable), producing a good output Q



- The synchronizer fails if D2 **has not resolved** to a valid level by the time $t_{res} > T_C - t_{setup}$
- The probability of failure is
  - $\text{Prob(failure)} = \frac{T_0}{T_C} \cdot \exp\left(-\frac{T_C - t_{\text{setup}}}{\tau}\right)$
- If D changes N times per second, the probability of failure per second is
  - Prob(failure)/sec = N * Prob(failure)



System reliability

- We can define the system reliability as as the **mean time between failures** (MTBF)
  - the reciprocal of the probability that the system will fail in any given second
  - $$\mathrm{MTBF} = \frac{1}{\mathrm{Prob(failure)/sec}} = \frac{T_C}{NT_0} \cdot \exp\left(\frac{T_C - t_{\mathrm{setup}}}{\tau}\right)$$
- MTBF improves exponentially as the synchronizer waits for a longer time
  - for most systems, to wait for one clock cycle provides a safe MTBF
  - in exceptionally high-speed systems, waiting for more cycles may be necessary
- A system receives asynchronous inputs from a sensors on average 0.2 times per second. How long must be the clock period for the MTBF to exceed 1 year?
  - flip-flops in the synchronizer: τ=200ps, T_0=150ps, t_{setup}=500ps
  - MTBF = 1y = 3.14 10^7 sec
  - $$3.14 \times 10^7\,\mathrm{s} = \frac{T_C}{0.2 \cdot 150 \times 10^{-12}} \cdot \exp\left(\frac{T_C - 500 \times 10^{-12}}{200 \times 10^{-12}}\right)$$
  - no closed form solution, however, it is easy enough to solve by guess and check
  - Tc = 3.036 ns