Static Timing Analysis

Part 3

Amr Adel Mohammady

/amradelm

This Document Is Dedicated to Thousands of Palestinian Children Who

Were Killed

Lost Their Limps

Became Orphans

Are Starved

At The Hands of These War Criminals

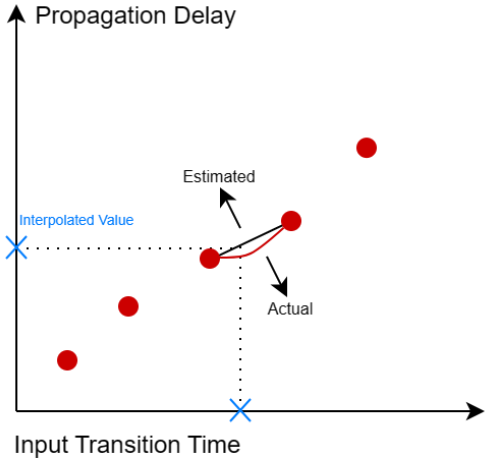
Introduction

- In part 1 we went through the basic principles that are needed to understand all VLSI timing checks.
- In part 2 we looked into setup and hold checks.
- In this parts we will go through other checks.
- **The timing checks covered in this part are:**
  - Max transition
  - Max load capacitance
  - Min pulse width
  - Max and min delays
  - Skew

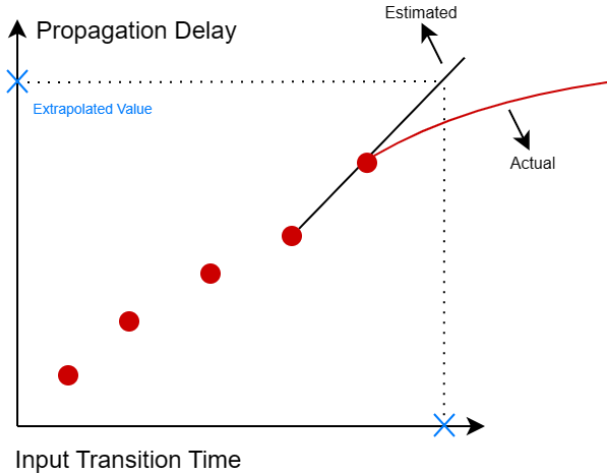
Max Transition

Interpolation vs Extrapolation

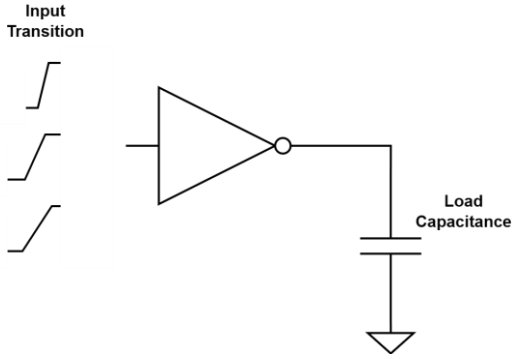
- In part 1 we talked about timing tables and showed how they are used to calculate the cell transition and propagation delays
- We also showed that if the input doesn't exist in the table, we assume linear connection between 2 known points and calculate the required delay.
- When the required value lies between two known values we call this **interpolation**. If the required value lies outside the known values we call this **extrapolation**
- In interpolation, the difference (error) between the estimated and the actual values is small if we have enough samples. But in the case of extrapolation, the difference could be large as we are estimating in an unknown region.
- This means we can't rely on extrapolated values during our static timing analysis. So, we will apply a limit that the max input transition for each cell shouldn't exceed the largest value in the timing table. The value of this limit is defined in the timing libraries for each cell



Interpolation



Extrapolation



Input Transition Time
t

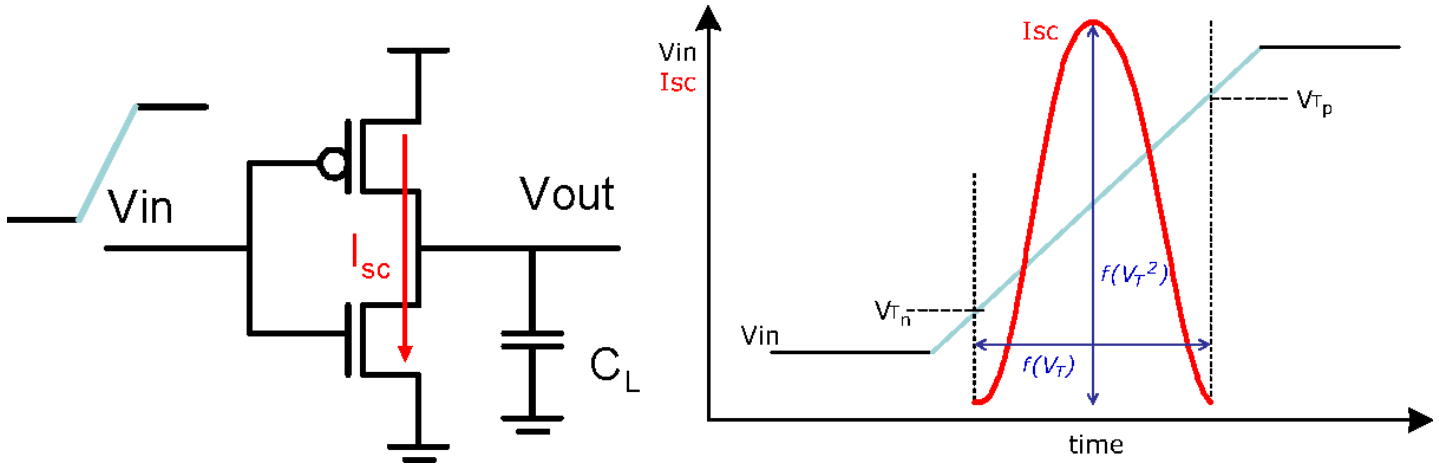
Load Capacitance
C_L

	1.1	1.2	1.3	1.4
10	2.10	2.20	2.27	3.00
20	2.50	3.00	3.45	3.96
30	2.90	3.40	3.80	4.15

Example Propagation Delay Timing Table

Short Circuit Power

- Another reason to limit the maximum transition is power consumption.
- During the transition of a logic gate, both the NMOS and PMOS networks are short circuit for a small amount of time. This causes a big current to flow from VDD to GND. The power consumed due to this current is called the **short circuit power**.
- To reduce this power component, we need to reduce the transition time. Therefore, we might apply another max transition constraint on the design along with the one already defined for each cell.
- The constraint shouldn't be too tight or the upsizing and buffer insertion (see next slide) to fix the violations will lead to more power consumption that exceeds what we wanted to saved.

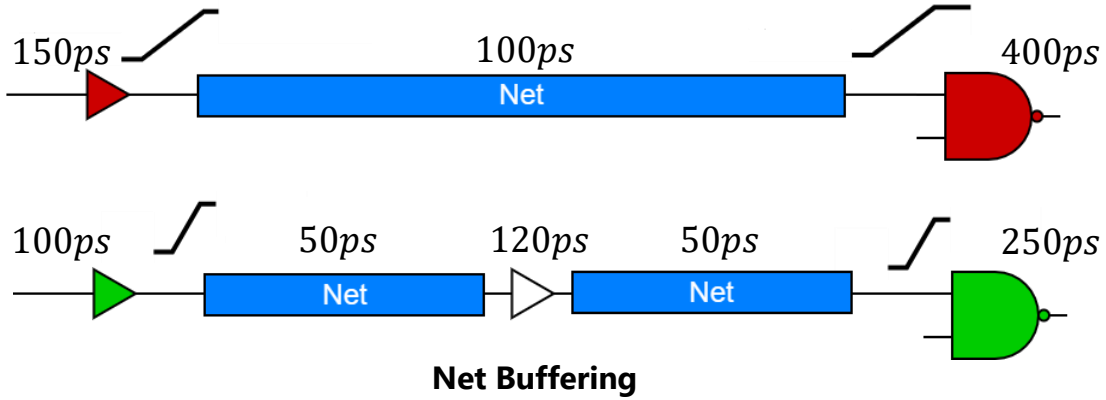


Short Circuit Current

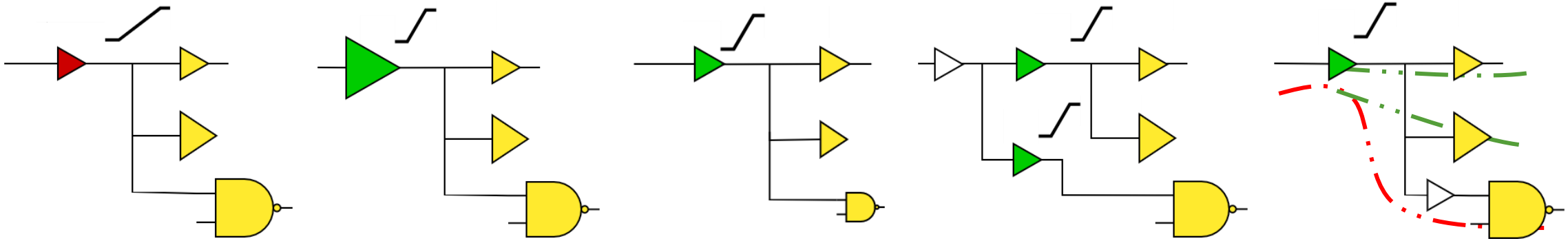
Reference : Morgenshtein, Arkadiy. "Short-Circuit Power Reduction by Using High-Threshold Transistors." Journal of Low Power Electronics and Applications 2 (2012): 69-78.

How to Fix a Max Transition Violation

- In part 2 we showed several methods to enhance the driving strength which in turn reduces the transition time. Such methods are net buffering[1], upsizing the driver, reducing the load cap, etc.



Net Buffering



Original | **Upsizing the driver** | **Downsizing the load** | **Fanout splitting** | **Side Load Isolation**

[1]: PNR engineers sometimes use max_transition constraints to force the tool to break up and buffer long nets

Max Capacitance

Max Capacitance

- Like max transition, we can't rely on extrapolated values when we calculate the delay due to the load capacitance. The timing libraries define a max cap value for each cell
- Sometimes the design contains analog circuits that require limits on the load capacitance on the pins. We need to manually apply max capacitance constraints for them[1]

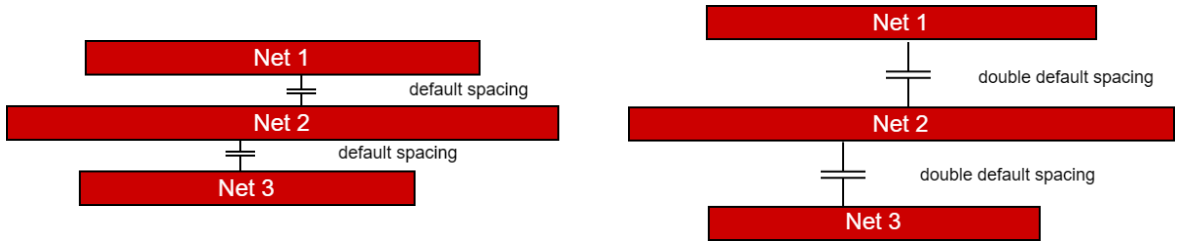
```
max_capacitance : 0.5026340000;
max_transition : 1.5045660000;
power_down_function : "(!VPWR + VGND)";
related_ground_pin : "VGND";
related_power_pin : "VPWR";
timing () {
    cell_fall ("del_1_7_7") {
        index_1("0.0100000000, 0.0230506000, 0.0531329000, 0.1224740000, 0.2823110000, 0.6507430000, 1.5000000000");
        index_2("0.0005000000, 0.0015825200, 0.0050087700, 0.0158530000, 0.0501755000, 0.1588080000, 0.5026340000");
        values("0.1812610000, 0.1852033000, 0.1952684000, 0.2175679000, 0.2638536000, 0.3705279000, 0.6698899000", \
            "0.1858589000, 0.1897823000, 0.1998833000, 0.2222378000, 0.2686923000, 0.3753406000, 0.6750271000", \
            "0.1962532000, 0.2001951000, 0.2102637000, 0.2325911000, 0.2788858000, 0.3855981000, 0.6848581000", \
            "0.2139762000, 0.2179139000, 0.2279842000, 0.2501680000, 0.2967764000, 0.4034162000, 0.7032850000", \
            "0.2347944000, 0.2386233000, 0.2485787000, 0.2708004000, 0.3172033000, 0.4242022000, 0.7234984000", \
            "0.2528099000, 0.2567236000, 0.2667510000, 0.2890031000, 0.3355522000, 0.4421618000, 0.7417389000", \
            "0.2441395000, 0.2480549000, 0.2581288000, 0.2803808000, 0.3269595000, 0.4338373000, 0.7336718000");
    }
    cell_rise ("del_1_7_7") {
        index_1("0.0100000000, 0.0230506000, 0.0531329000, 0.1224740000, 0.2823110000, 0.6507430000, 1.5000000000");
        index_2("0.0005000000, 0.0015825200, 0.0050087700, 0.0158530000, 0.0501755000, 0.1588080000, 0.5026340000");
        values("0.1824226000, 0.1860255000, 0.1956822000, 0.2213225000, 0.2947436000, 0.5205480000, 1.2368203000", \
            "0.1867734000, 0.1903485000, 0.2000419000, 0.2256768000, 0.2991186000, 0.5247163000, 1.2371056000", \
            "0.1990939000, 0.2026934000, 0.2123735000, 0.2380258000, 0.3115148000, 0.5372138000, 1.2494737000", \
            "0.2264985000, 0.2300219000, 0.2397595000, 0.2653641000, 0.3388854000, 0.5646577000, 1.2771347000", \
            "0.2841544000, 0.2877657000, 0.2974600000, 0.3231024000, 0.3965533000, 0.6222400000, 1.3348850000", \
            "0.3865151000, 0.3901513000, 0.4000870000, 0.4259103000, 0.4993993000, 0.7254375000, 1.4385320000", \
            "0.5620023000, 0.5660107000, 0.5763886000, 0.6025792000, 0.6765027000, 0.9025882000, 1.6156780000");
    }
```

Timing Library from Skywater 130nm Open-source PDK

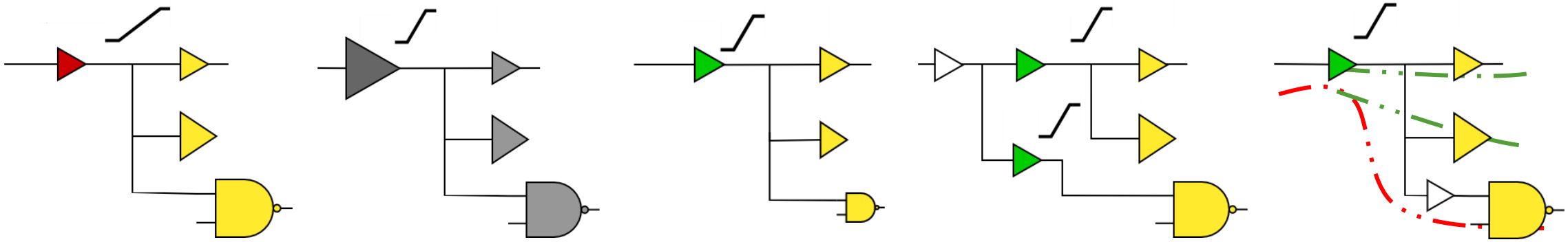
[1]: Some analog blocks have timing libraries that contain the max_transition and max_capacitance limits

How to Fix a Max Capacitance Violation

- In part 2 we showed several methods to reduce the load capacitance such as downsizing the load, reducing the coupling capacitance between wires with spacing, fanout splitting, side load isolation, etc
- **Note that upsizing the driver won't fix the violation because:**
  - The load capacitance won't decrease with upsizing. In fact, upsizing will increase the internal parasitic capacitance of the driver and thus increase the overall capacitance
  - The only way upsizing could fix the violation is if the timing tables of that bigger cell is characterized at a larger load capacitance range hence no extrapolation occurs but that's not always the case.



Wire Spacing

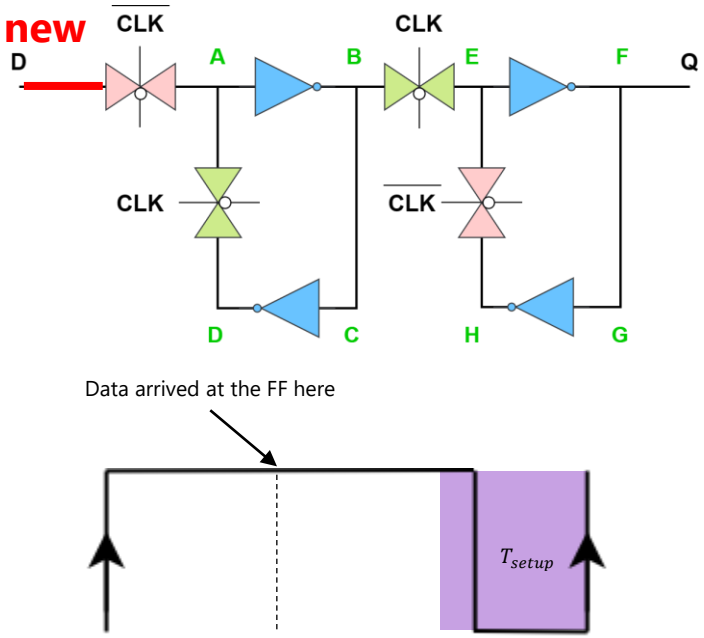


Original | ~~**Upsizing the driver**~~ | **Downsizing the load** | **Fanout splitting** | **Side Load Isolation**

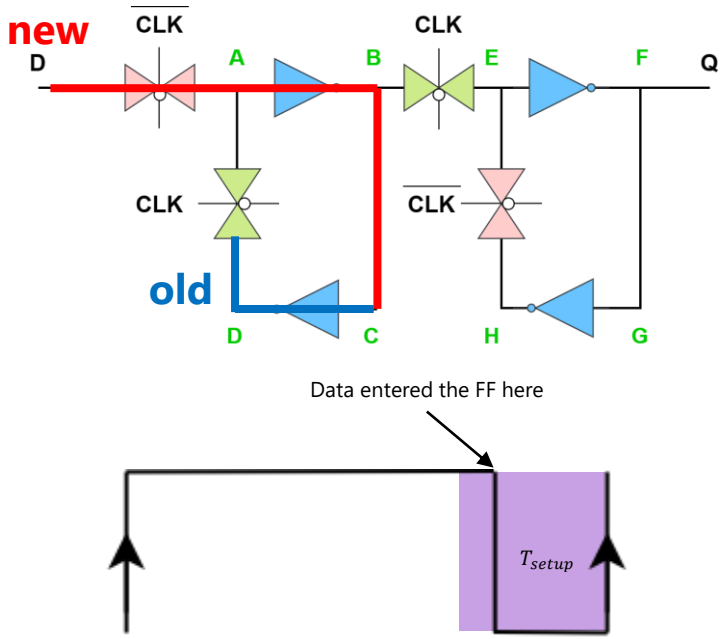
Min Pulse Width

Flip Flop Internal Operation

1 To understand the min pulse width check we need to have a look at the internal operations of a FF.
Lets assume **new** data arrived at the D pin of a positive edge triggered FF before the setup time but during the high level of the clock.
The data can't enter the FF because the red transmission gates are still open circuit

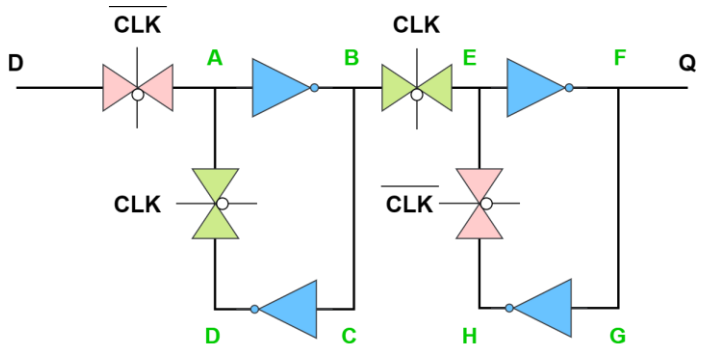


2 When the clock becomes low level (CLK=0) the red transmission gates become short circuit.
The **new** data enters the FF and starts overwriting the **old** stored value
However, the clock edge arrives before the **new** data overwrites all the internal nodes of the FF. We have two conflicting values inside the FF and metastability occurs



Flip Flop Internal Operation

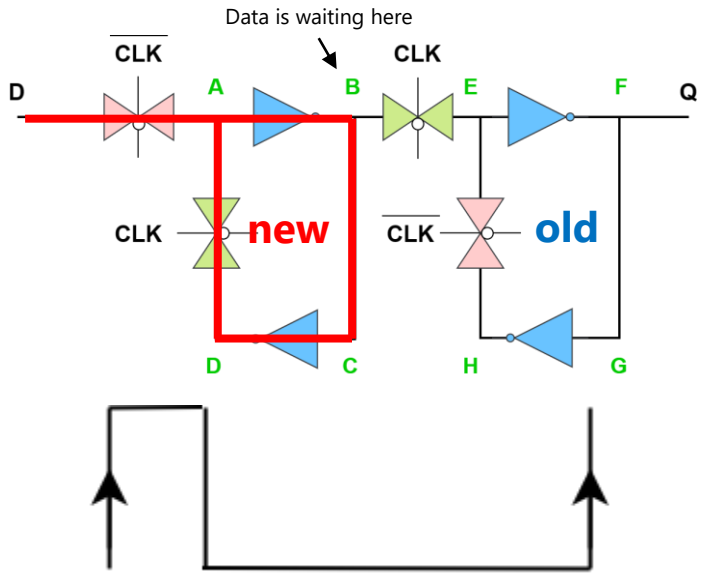
- Although the new data arrived before the setup time, metastability still occurred because the low level width was too small.
- We need to make sure the width is wide enough to allow the data to enter the FF and overwrites the internal nodes
- The low level width should be equal or larger than the FF setup time.



Reference : https://www.design-reuse.com/articles/37652/delay-characterization-for-sequential-cell.html

Flip Flop Internal Operation

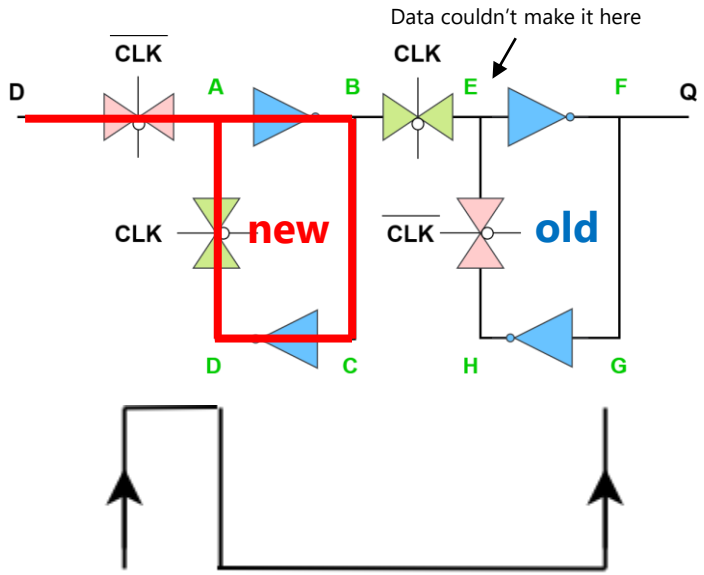
1 Lets consider the opposite case where the high level edge is small.
The **new** data entered the FF and overwrote the internal nodes and is waiting for the clock edge (CLK=1) to come so that it can reach the Q pin



2 The clock edge comes (CLK=1). The green transmission gates start to switch and become short circuit.

The **new** data is waiting for the green gate to finish switching and become short circuit in order to cross from **B to E** and then to Q.

Because the high level edge is small, the low level (CLK=0) comes before the green gate switch. The new data couldn't reach Q and wasn't captured.



Flip Flop Internal Operation

3 Lets consider another case, the high level pulse is small but wide enough that the new data crossed **B-E-F** and then to Q.

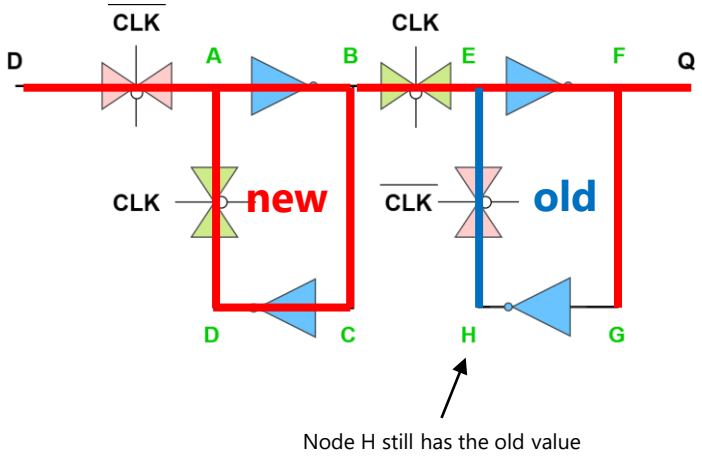
The clock becomes low (CLK=0) before the new data overwrites node **H.**

The red transmission gates are now short circuit and nodes **E,H** and connected to each other. There are 2 values conflicting inside the FF.

So, metastability will occur.

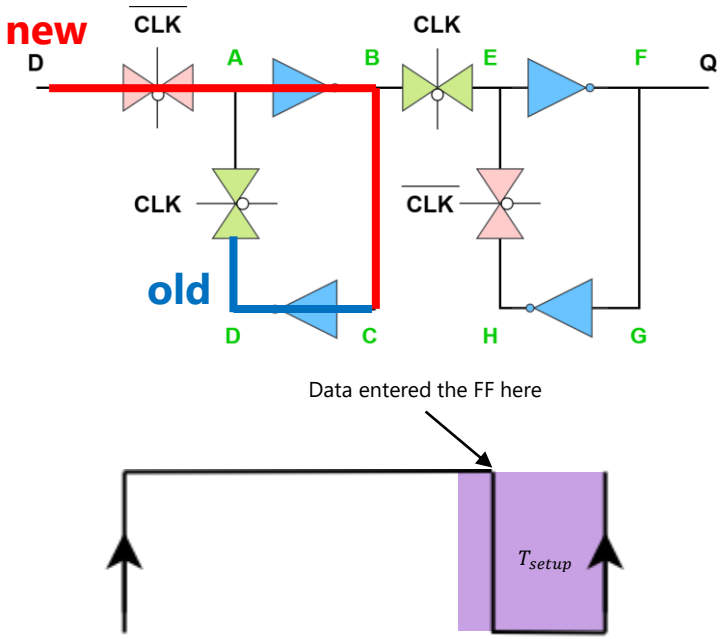
We need to make sure the high level width is wide enough to allow the data to cross from **B to E** and also wide enough to overwrites the internal nodes in the 2nd inverter loop

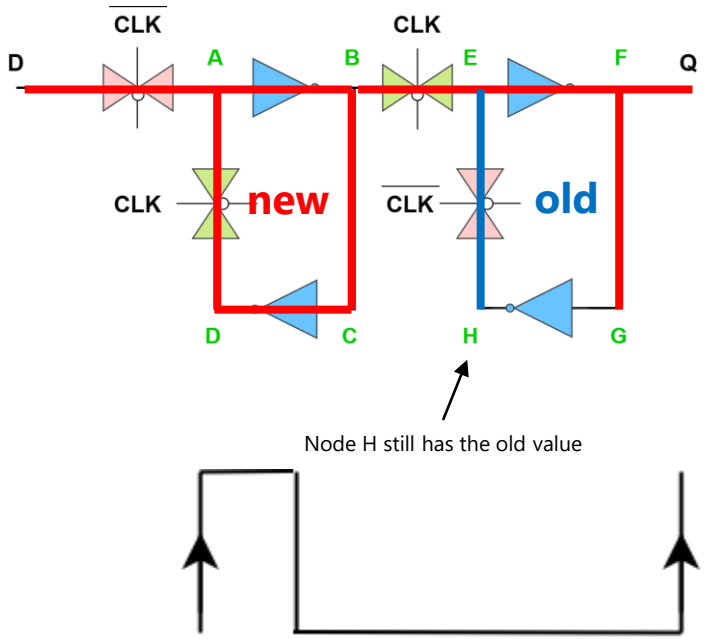
The high level width should be larger than the delay from **B** through **E-F-G-H**. This delay is comparable to the setup delay time (D pin to **A-B-C-D**)



Flip Flop Internal Operation

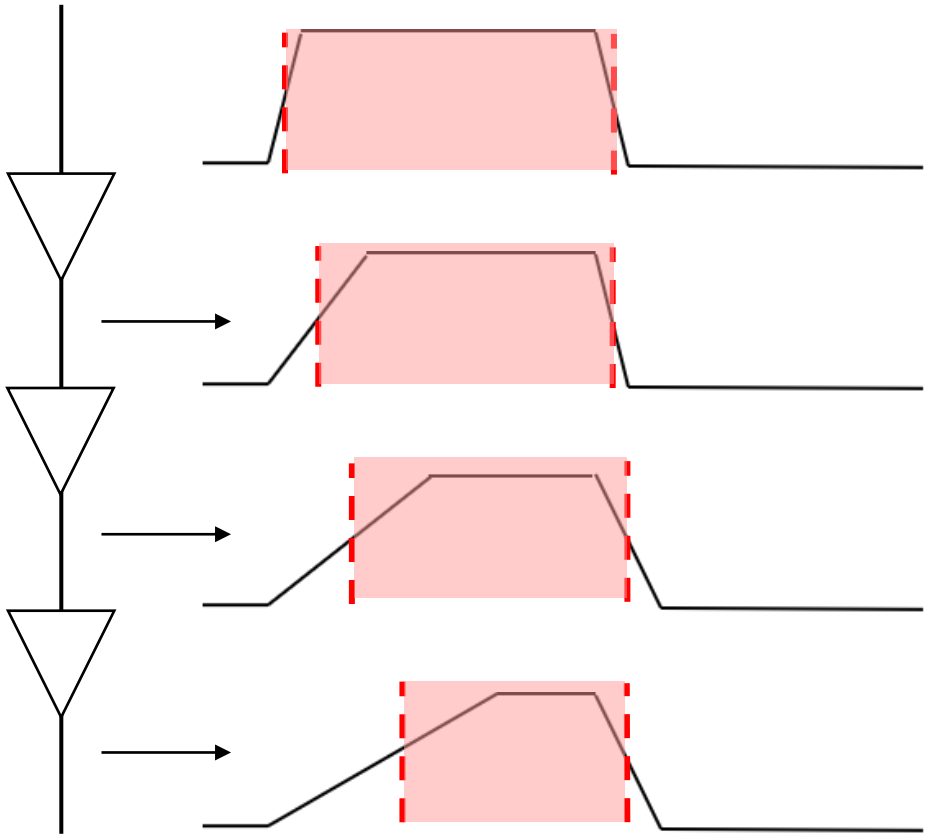
- The previous examples show that while flip-flops are edge triggered, the width of the pulses is important to guarantee correct operation of the FFs
- The min pulse width limits are defined inside the timing library for each sequential cell
- Also some blocks such as memory blocks or analog blocks may require a limit for the pulse width of some input signals.





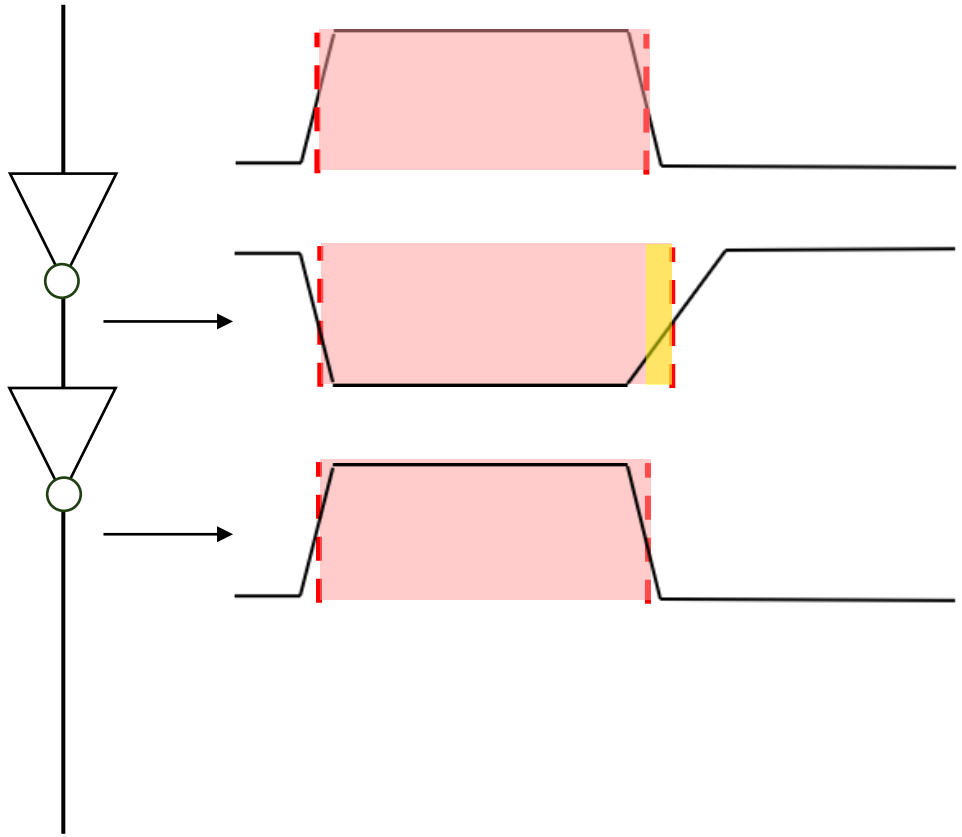
What Distorts the Pulse Width?

- Pulse distortion occurs due to the difference between the rise time and fall time of the logic cells
- **The example in the left shows a pulse going through a chain of buffers**
  - Lets assume the rise time of the buffers is slower than the fall time.
  - The 1st buffer produces **slow** 1st edge (rise) and **fast** 2nd edge (fall)
  - The 2nd buffer sees a slower rise time than the one seen by the 1st buffer, and so produces a much **slower** 1st edge (rise) and **fast** 2nd edge (fall)
  - The same case happens with the 3rd buffer resulting in a distorted signal.



How to Fix a Min Pulse Width Violation

- **To avoid distortion use inverter pair with the same size instead of buffers[1]:**
  - Lets assume the inverters has slower rise time.
  - The 1st inverter produces **fast** 1st edge (fall) and **slow** 2nd edge (rise)
  - The 2nd inverter produces **slow** 1st edge (rise) and **fast** 2nd edge (fall)
  - The inverters will cancel the effects of each other. The key here is that the polarity of the pulse edges will be inverted, what was a fall becomes a rise and what was a rise becomes a fall.
- Also, some standard cell libraries contain different versions of the same cell with different PMOS and NMOS transistor sizes to get different transition times (symmetric rise and fall, more rise, more fall) to aid the backend designer in fixing min pulse width violations.

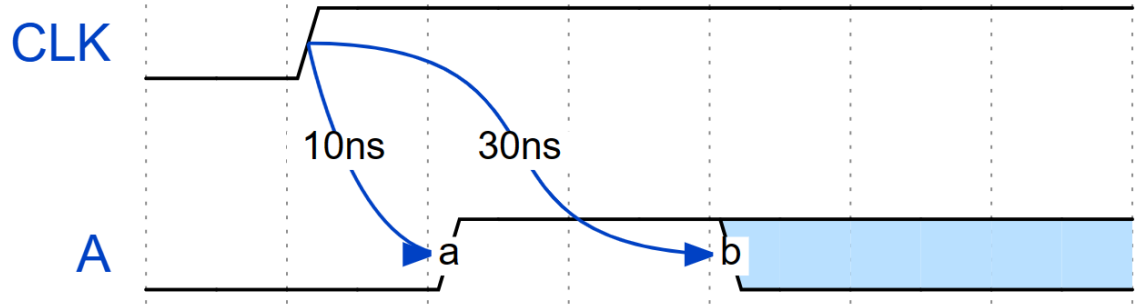


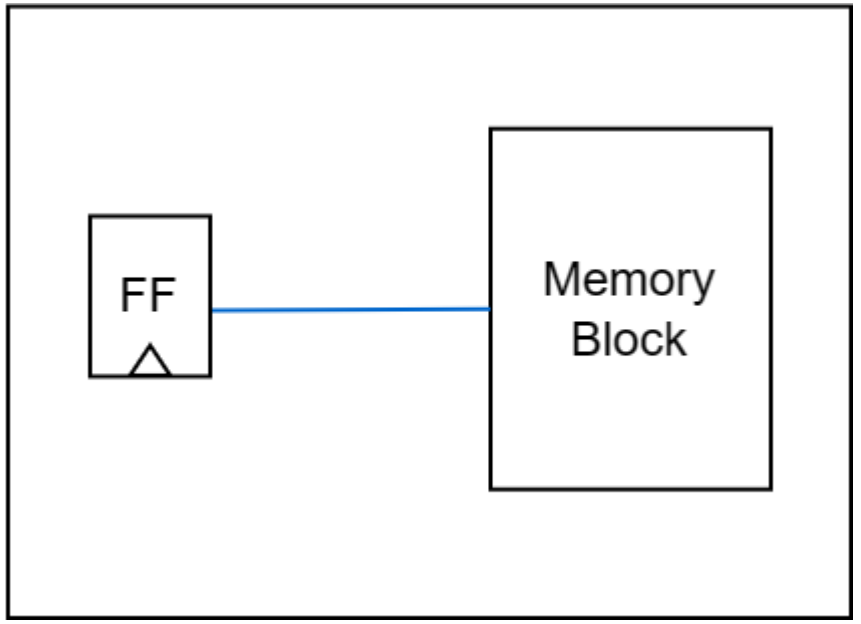
[1]: Although buffers are internally made of 2 inverters connected back to back, the sizes and loads of the inverters are different leading to different rise and fall times. More details here : https://www.physicaldesign4u.com/2020/03/cts-part-iii-clock-buffer-and-minimum.html

Max and Min Delays

Max and Min Delays

- Sometimes we want to control the arrival time of a signal.
- In the example below, it's required that signal A arrives at the memory block no eariler than 10ns and no later than 30ns after the clock edge.
- To constraint signal A to follow this requirement we need to apply a min delay constraint of 10ns and a max delay of 30ns[1].
- Max and min delays are solved by speeding up or slowing down the delays using the methods we discussed with setup and hold violations.





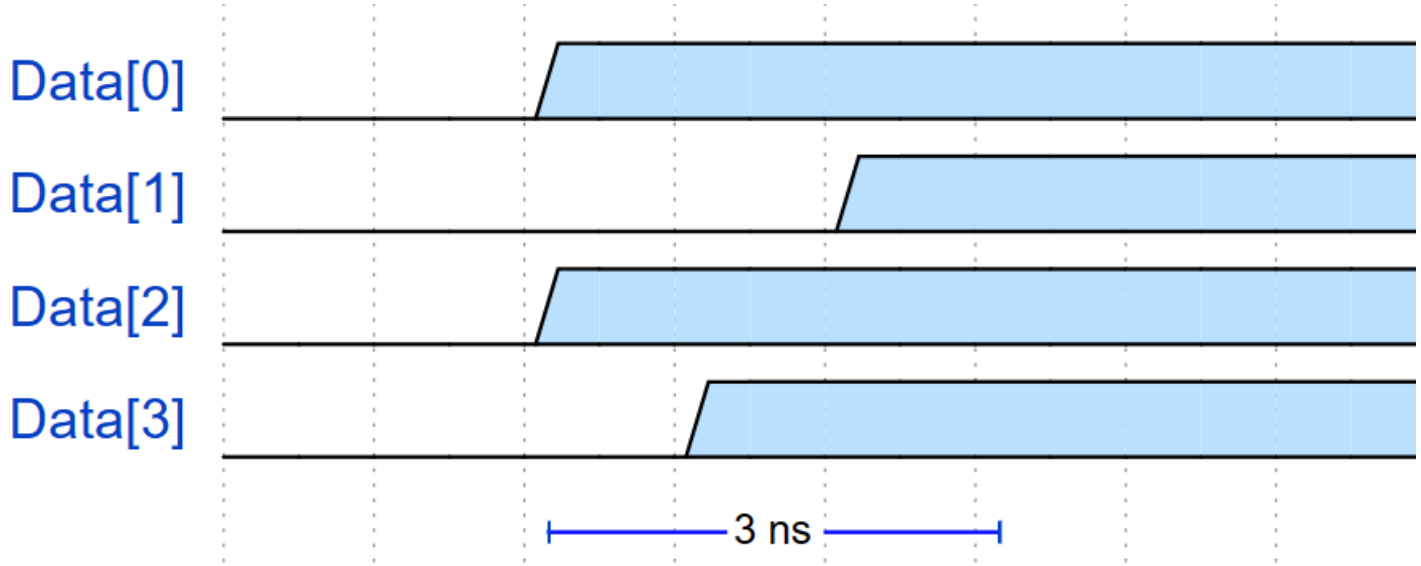
More details : https://docs.amd.com/r/2021.2-English/ug903-vivado-using-constraints/Min/Max-Delays

[1] : Don't apply the constraint from the Q pin of the FF but from the CK pin. Otherwise, the setup and hold timing paths of the FF will be broken

Skew

Skew

- Skew checks constraint the arrival difference between 2 signals or more.
- In the example below we have a data bus of 4 bits. The bits should arrive close to each other with a difference no more than 3ns. This means the difference between the latest bit to arrive and the earliest bit to arrive shouldn't exceed 3ns.
- To fix skew violations we need to speed up slow signals and/or slow down fast signals using the methods we discussed before to decrease or increase the delay



Thank You!