Assignment on Architectural Design of Digital Integrated Circuits

Number-9

(Each question carries 10 marks each)

1. **A) Explain Positive, negative and zero setup time.**
   B) Explain Positive, negative and zero hold time.

A) Solution: As we know from the definition of setup time, setup time is a point on time axis which restrains data from changing after it. Data can change only before occurrence of setup timing point. Theoretically, there is no constraint on occurrence of setup time point with respect to clock active edge. It can either be before, after or at the same time as that of clock edge. Depending upon the relative occurrence of setup time point and clock active edge, setup time is said to be positive, zero or negative.

Positive setup time: When setup time point is before the arrival of clock edge, setup time is said to be positive. Figure 1 below shows positive setup time.

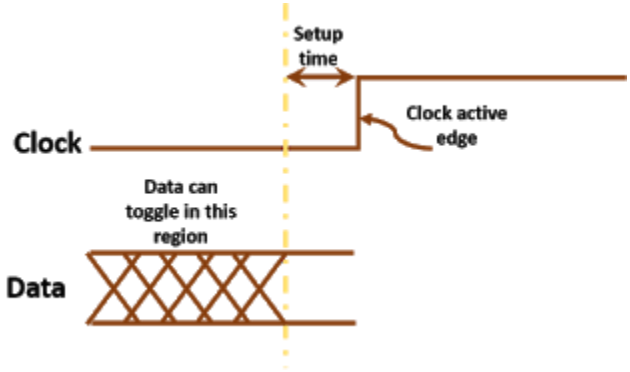


Figure 1: Positive setup time

Zero setup time: When setup time point is at the same instant as clock's active edge, setup time is said to be zero. Figure 2 shows a situation wherein setup time is zero.

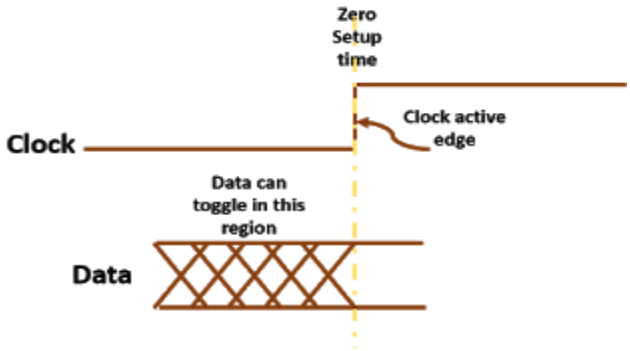


Figure 2: Zero setup time

Negative setup time: When setup time point occurs after clock edge, setup time is said to be negative. Figure 3 shows timing waveform for negative setup time.

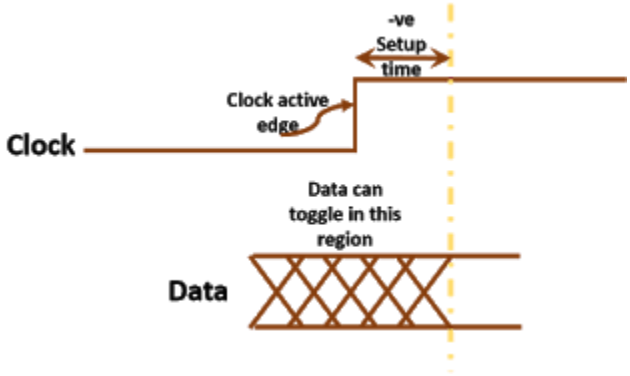


Figure 3: Negative setup time

What causes different values of setup time: Figure 4 shows a positive level-sensitive D-latch. As we know from the definition of setup time, setup time depends upon the relative arrival times of data and clock at input transmission gate (We have to ensure data has reached upto Node D when clock reaches input transmission gate). Depending upon the relative arrival times of data and clock, setup time can be positive, zero or negative.

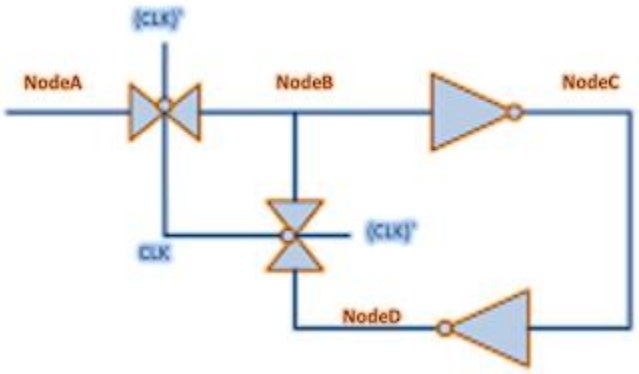


Figure 4: Positive level-sensitive latch

Let us assume the delay of an inverter is 1 ns. Then, to ensure that the data has reached Node D when clock edge arrives at input transmission gate, data has to be available at the input transmission gate at least 2 ns before. So, if both data and clock reach the reference point at the same time, the latch has a setup time of 2 ns.

Now, if data takes 1 ns more than clock to reach input transmission gate from the reference point, then, data has to reach reference point at least 3 ns before clock reference point. In this case, setup time will be 3 ns. Similarly, if data takes 1 ns less than clock to reach input transmission gate, setup time will be 1 ns. And if data takes 2 ns less than clock to reach input transmission gate, setup time will be zero. Now, if there is further difference between delays of data and clock from respective reference points to input transmission gate, the hold time will become negative. For example, if data takes 3 ns less than clock to reach input transmission gate, setup time will be -1 ns.

This is how setup time depends upon relative delays of data and clock within the sequential element. And it completely makes sense to have negative setup time.

B) Solution: As we know from the definition of hold time, hold time is a point on time axis which restrains data from changing before it. Data can change only after hold time has elapsed. Now, there is no constraint on the occurrence of hold time point with respect to clock edge. It can either be after, before or at the same instant of time as that of clock active edge.

Posotive hold time: When hold time point is after the arrival of clock active edge, hold time is said to be positive hold time. Figure 1 below shows positive hold time.

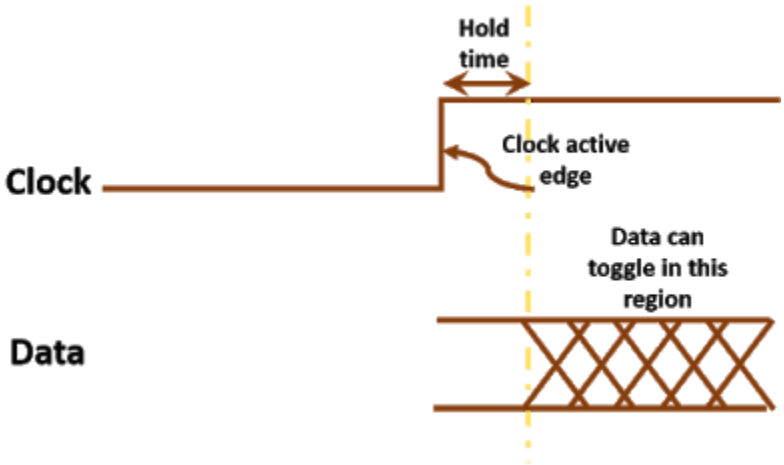


Figure 1: Positive hold time

Zero hold time: When hold time point is at the same time instant as that of clock active edge, we say that

hold time of the sequential element is zero. Figure 2 below shows timing waveform for zero hold time.

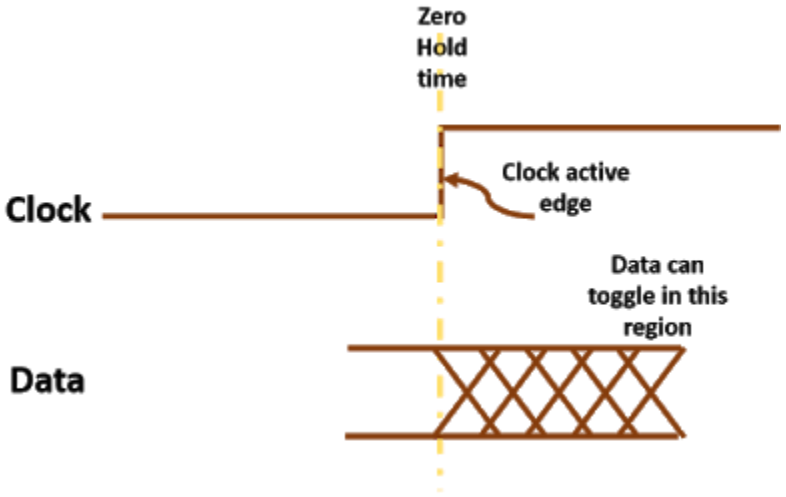


Figure 2: Zero hold time

Negative hold time: Similarly, when hold time point comes earlier on time scale as compared to data, we say that hold time of the sequential element is negative. Figure 3 shows timing waveform for negative hold time.

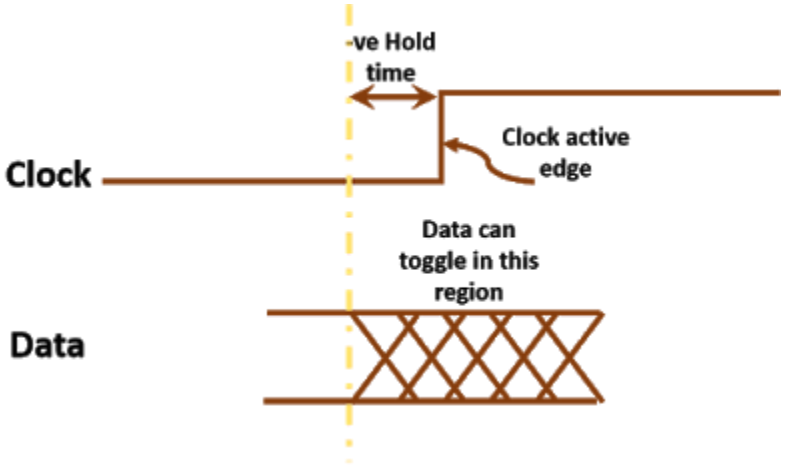


Figure 3: Negative hold time

Figure 4 shows a positive level-sensitive D-latch. As we know (from definition of hold time), hold time depends upon the relative arrival times of clock and data at the input transmission gate (We have to ensure data does not reach Node C). Depending upon the times of arrival of clock and data, hold time can be positive or negative.

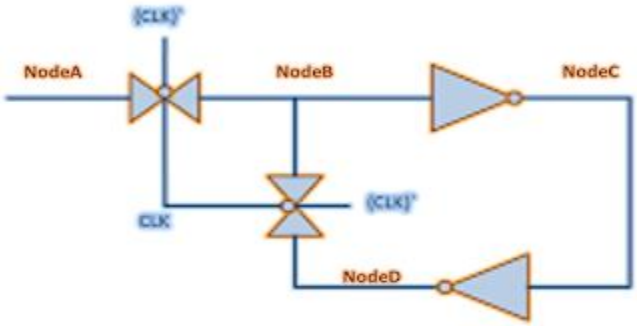
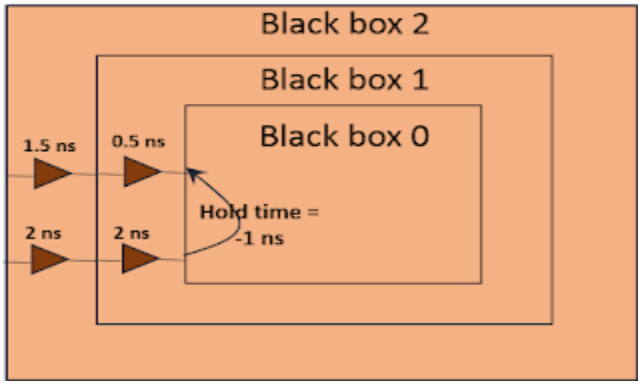


Figure 4: Positive level-sensitive D-latch

Let us say, the delay of an inverter is 1 ns. Then, we can afford the data to reach transmission gate input even 0.9 ns before arrival of clock at transmission gate. This will ensure data reaches Node C (-0.9 + 1 =) 0.1 n after arrival of clock edge, if allowed. But, since, clock closes transmission gate, data will not reach Node C. So, in this case, hold time is -1 ns. If the delay from Node B to Node C was something else, hold time would also have been different.

Now, if we say that clock arrives at transmission gate 1 ns earlier than data, then, by above logic, hold time of this latch will be -2 ns. Similarly, if clock arrives at transmission gate 0.5 ns after data, hold time will be -0.5 ns. And if clock arrive at transmission gate 1 ns after data, hold time will be zero. If the arrival time of clock is made more lately, hold time will be greater than zero. For example, if arrival time of clock is 2 ns after data, hold time will be +1 ns.

Hold time of the circuit is also dependent upon the reference point. For example, consider a multi-level black box as shown in figure 5. If we look at black box 0, its hold time is -1 ns. At level of black box 1, wherein clock travels 2 ns and data travels 0.5 ns to reach black box 0, hold time is (-1 + 2 - 0.5 =) 0.5 ns. Similarly, at the level of black box 2, hold time is 1 ns. This is how; hold time depends upon the relative arrival times of clock and data. And it completely makes sense to have a negative hold time.



2. **A) How positive edge trigger register to positive latch path is zero cycle. But positive latch to rising flop is full cycle?**

B) How can we generate a pulse for every edge of the incoming pulse?

A) Solution:

Why setup check for postive latch to positive edge-triggered register is full cycle: Figure 1 below shows the clock waveforms for a positive latch to positive edge-triggered flip-flop timing paths. Positive edge-triggered register samples data only on positive edges. In the below figure, those instances are either "**Time = 0**" or "**Time = T**". The output of latch can change anytime when it is transparent. The earliest it can change is at "**Time = 0+**". So, the next instant it can get captured at the register is "**Time = T**". This makes the default setup check for such paths. That is why setup check for positive latch to positive edge-triggered registers is full cycle.

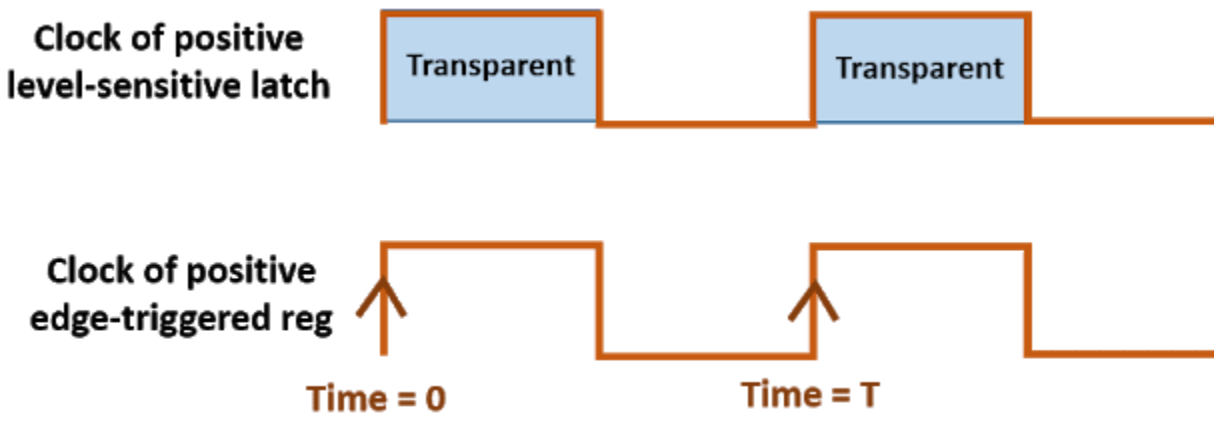


Figure 1: Single cycle setup check from positive latch to positive edge-triggered flip-flop

Why setup check for positive edge-triggered register to positive level-sensitive latch: Similar to the above case, figure 2 shows the clock wave-forms for positive edge-triggered flip-flop to positive latch timing paths. The flip-flop can launch data at either "**Time = 0**" or "**Time = T**". So, data will be available at its output at either "**Time = 0+**" or "**Time = T+**". The latch can capture the data at the same instant as it launched as it is transparent at that time. So, setup check is considered to be zero cycle in this case.

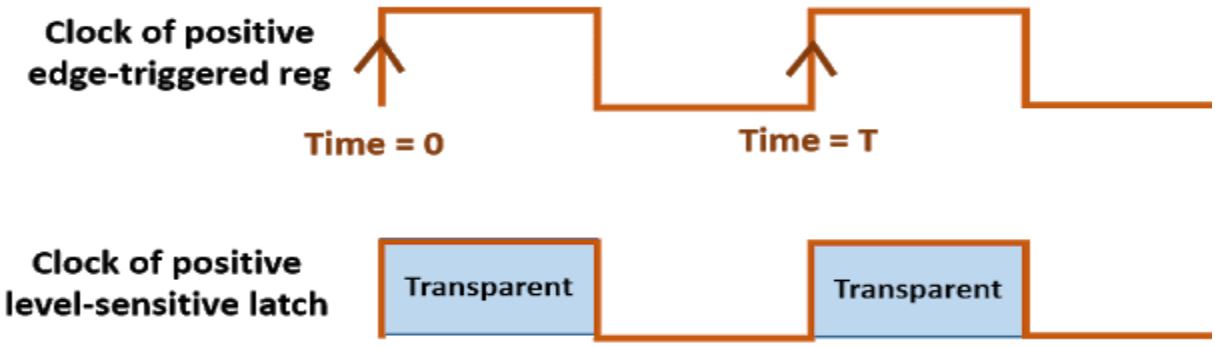


Figure 2: Zero cycle setup check from positive edge-triggered flip-flop to positive latch

We need to note that the "**from**" and "**to**" edges in these checks denoted in text-books (or shown in timing reports by STA tools by default) are default setup and hold checks. The actual setup and hold check edges are what is represented by the state machine the timing path belongs to. It is possible to override default check edges by command **set_multicycle_path**.

B) Solution: It is a very common requirement to detect either positive edge, negative edge or both edges of a signal. And the circuit that can detect and generate a single cycle pulse is quite simple. Here, we will discuss how we can detect positive edge, negative edge and both edges of a signal.

- **Detect positive edge of a signal**: Positive edge of a signal means that the current state of the signal is "1" and previous state is "0". And a pulse signal means that the output of the circuit is "1" for one cycle. So, we need a circuit which generates "1" as output when present state is "1" and previous state is "0". It generates "0" as output otherwise.

Thus, output = D(n-2)' & D(n-1)

The required implementation is shown in figure 1 below:

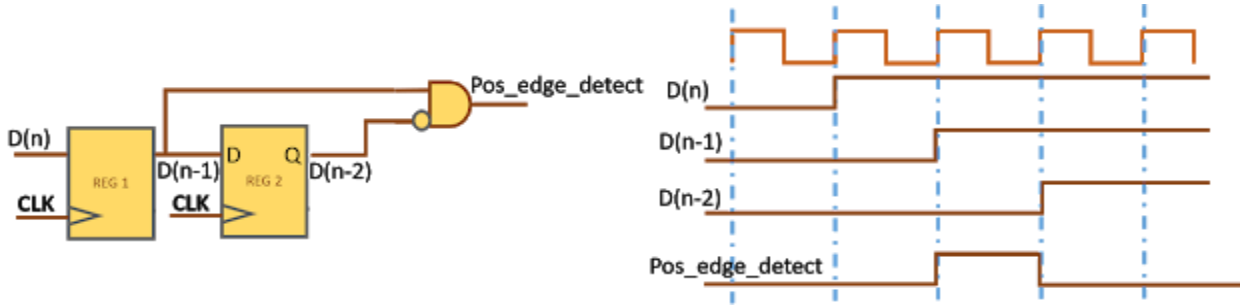


Figure 1: Detection of positive edge of signal

- **Detect negative edge of a signal**: Negative edge of signal means that the current state of the signal is "0" and previous state is "0". So, we need a circuit which generates "1" as

output when present state is "0" and previous state is "1". It generates "0" as output otherwise.

Thus, Neg_edge_detect = D(n-2) & D(n-1)'

The required implementation is shown in figure 2 below:

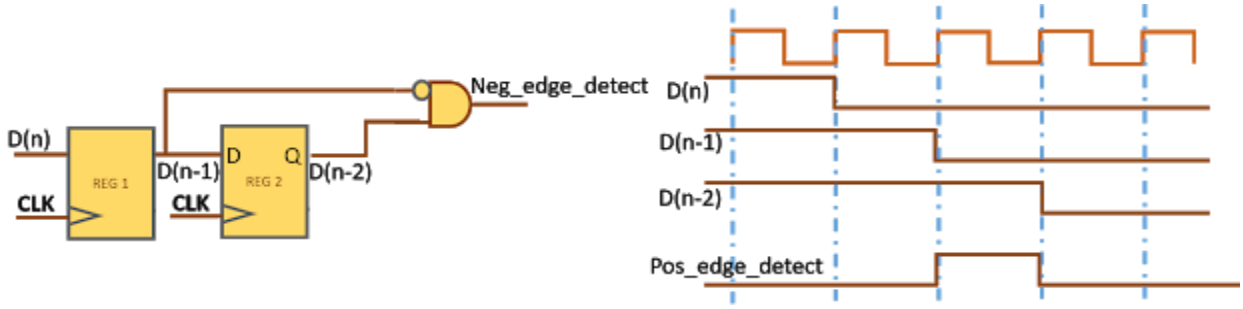


Figure 2: Detection of negative edge of signal

- **Detecting both positive and negative edges of the signal**: Simply doing an "OR" operation of Pos_edge_detect and Neg_edge_detect signal will produce an output which is a single cycle pulse for any of the edge of incoming signal. The requirement for consecutive edges of incoming signal is to be at least 2 cycles apart otherwise, the output will not be pulse, but will be a continuous signal.

Any_edge_detect = Pos_edge_detect + Neg_edge_detect

The required implementation is shown in figure 3 below:

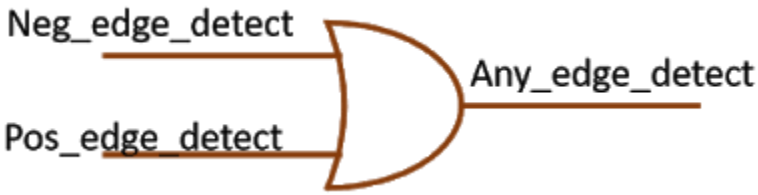


Figure 3: Detection of both positive and negative edges of signal

3. How a latch/flip-flop goes metastable?

Solution: Since, latches and flip-flops consist of inverter loops controlled by transmission gates; they also are susceptible to meta-stability. For instance, consider a negative latch as shown in figure 1 and the clock waveform alongside. The instance of interest to us is the instance when switch_1 closes or at the transparency_close edge of the latch. Also, we know from theory two important concepts, "setup time" and "hold time". Let us call the region between setup time and hold time as setup-hold-window. If the data toggles before setup-hold-window, it is guaranteed to get captured and propagated to latch output. If data toggles after setup-hold-window, it is guaranteed not to get propagated to latch output. On the other hand,

if it toggles during setup-hold-window, it may or may not propagate to the output. Also, it may happen that when the switch closes, the input level is such that latch goes into metastable state.

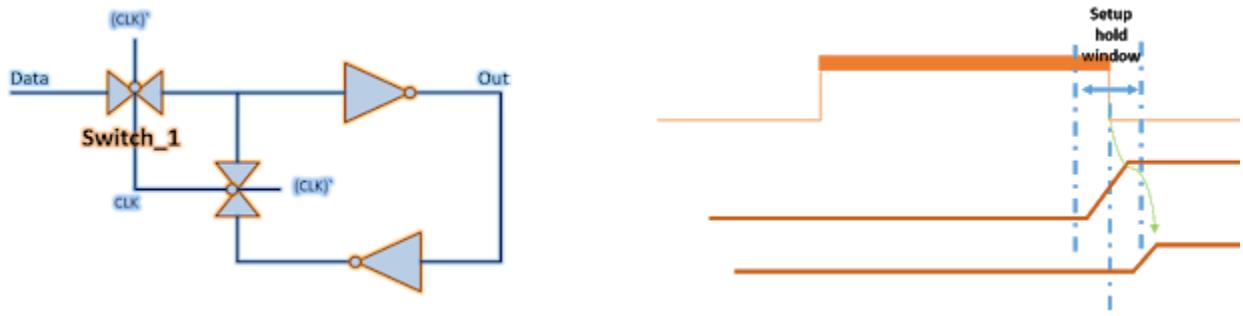


Figure 1: How latch goes into metastable state

As is evident from figure 1, input is still transitioning when switch has closed and the output goes metastable.

Similarly, as a flip-flop is also composed of latches configured in master-slave configuration, a flip-flop also goes metastable by same way. In general, we can describe it as: A flip-flop/latch has a defined timing requirement in terms of when data should be available at its input so that it is correctly captured. These requirements are termed as setup and hold times. If these requirements are not met, there is a possibility of flip-flop going metastable.

In general, following are the scenarios which can cause a flip-flop's output to go metastable.

- **Asynchronous timing paths**: Paths crossing clock domains, where the launch and capture clocks do not have definite phase relationship, cannot be assured to be captured outside setup/hold window.

- If there is a timing path violating setup and/or hold, then the capturing flip-flop will go metastable at a certain PVT, where it is probable to get captured in setup-hold-window

How metastability impacts design: Let us assume that the output of flip-flop goes to a number of gates (say 100). So, as long as the flip-flop is in metastable range, it will cause short circuit current to flow in all the gates. The short circuit current would be in the range of 100 uA. So, large amount of short circuit current will flow for a considerable amount of time.

What helps a flip-flop come out of metastability?

As described earlier, theoretically, it is possible for the flip-flop to remain in metastable state for infinite time in the absence of any disturbance. However, there are certain factors, which help it to come out of metastability.

- **Ability of the inverter pair to detect a disturbance and act on it**: If the inverter pair is able to detect even a smallest of the disturbances, it will act upon it and eventually come out of metastability. So, having these characteristics for transistors in inverter pair will help:
    - Low VT
    - High drive strength

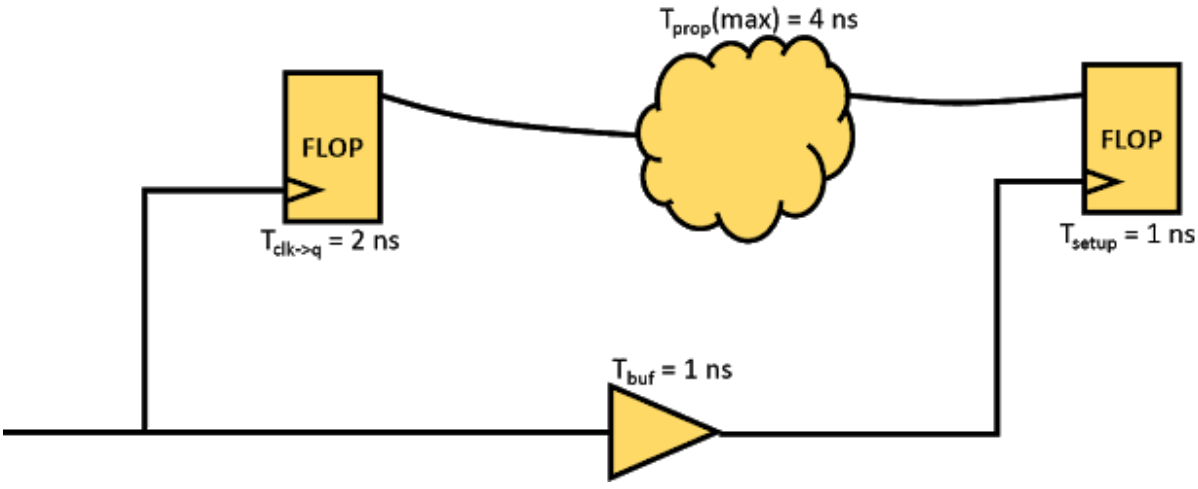
- Higher the time available for metastability resolution, more chances of having disturbance; hence, flip-flop will eventual come out of metastability

In general, the ability of a flip-flop to come out of metastability is measured by a parameter known as MTBF (Mean Time Between Failures). Lower the MTBF, higher the probability of flip-flop coming out of metastability within a given time. It depends upon:

- Technology factors
- Time available to resolve metastability: Higher the time available, lower if MTBF
- Frequency of the clock received by flip-flop: Higher the frequency of clock, higher is MTBF
- Frequency of toggling of data received by flip-flop: Higher is frequency of data, higher is MTBF
- Internal design of flip-flop: Ability of flip-flop to act on smallest of disturbances, as discussed earlier. In general, a flip-flop consuming more power and having high gain will be able to come out of metastability quickly

.

4. Figure below shows a timing path from a positive edge-triggered register to a positive edge-triggered register. Can you figure out the maximum frequency of operation for this path?



Solution: The above timing path is a single cycle timing path. The maximum frequency is governed by setup timing equation. In other words, maximum frequency of operation is the maximum frequency (minimum time period of clock) that satisfies the following condition:

$T_{ck->q} + T_{prop} + T_{setup} - T_{skew} < T_{period}$

Here,

$T_{ck->q}$ = 2 ns, T_{prop} = 4 ns, T_{setup} = 1 ns, T_{skew} = 1 ns, T_{period}

Now,

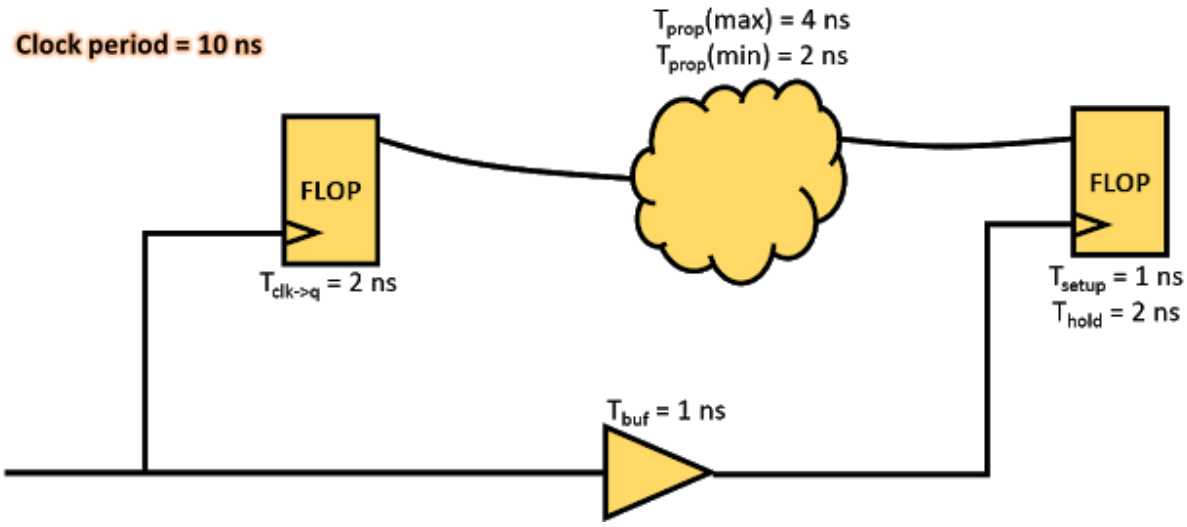
$T_{period} > 2 \text{ ns} + 4 \text{ ns} + 1 \text{ ns} - 1 \text{ ns}$

$T_{period} > 6 \text{ ns}$

So, the minimum time period of the clock required is 6 ns. And the maximum frequency that the above circuit can work is (1000/6) MHz = 166.67 MHz.

It should be noted that at if we operate this timing path at maximum frequency calculated, setup slack will be zero.

5. Figure below shows a timing path from positive edge-triggered register to a positive edge-triggered register. Can you figure out if there is any setup and/or hold violation in the following circuit?



Solution:

To check if a timing path violates setup and/or hold, we need to check if they satisfy setup and hold equations. A violating timing path has a negative setup/hold slack value.

The above circuit has a positive clock skew of 1 ns (as capture flip-flop gets clock 1 ns later than launch flip-flop).

Let us first check for setup violation. As we know, for a full cycle register-to-register timing path, setup equation is given as:

$T_{ck->q} + T_{prop} + T_{setup} - Tskew < T_{period}$

Here,

$T_{ck->q}$ = 2 ns, T_{prop} (max value of combinational propagation delay) = 4 ns, T_{setup} = 1 ns, T_{period} = 10 ns, Tskew = 1 ns

Now, $T_{ck->q} + T_{prop} + T_{setup} = 2 + 4 + 1 - 1 = 6$ ns $< T_{period}$

So, the above circuit does not have a setup violation. The setup slack, in this case, will be given as:

$SS = T_{period} - (T_{ck->q} + T_{prop} + T_{setup} - Tskew)$

SS = +4 ns

Since, setup slack comes out to be positive, this path does not have a setup violation.

Now, let us check is there is a hold violation for this timing path. Hold timing equation is given as:

$T_{ck->q} + T_{prop} > T_{hold} + T_{skew}$

Here,

$T_{ck->q}$ = 2 ns, T_{prop} (min value of combinational propagation delay) = 2 ns, T_{hold} = 2ns, T_{skew} = 1 ns

Now, $T_{ck->q} + T_{prop} = 2$ ns + 2 ns = 4 ns

And $T_{hold} + T_{skew} = 2$ ns + 1 ns = 3 ns

Now, 4 ns > 3 ns, so this circuit does not have a hold violation. The hold slack, in this case, will be given as:

$HS = T_{ck->q} + T_{prop} - (T_{hold} + T_{skew}) = +1$ ns

Since, hold slack comes out to be positive, this path does not have a hold violation.