Problem 1:

Without skew, there are two constraints that need to be met due to the loopback from L2 to the input of the first logic block and the loopback from L4 to the input of Δ1. In order to prevent infinite borrowing in the loopback that would cause data to arrive after L2 became opaque, it is necessary to make sure that any slack borrowing from FF → L2 is made up when the data loops back.

\[ \Delta 2 - w + \Delta 1 \leq w \quad \text{or} \quad 2w \geq \Delta 1 + \Delta 2 \quad \text{or} \quad T_{eye} \geq \Delta 1 + \Delta 2. \]

Consulting the timing diagram, we see that this means that:

\[ T_{eye} = 2w \]

This is true because \( \Delta 1 \) is always less than \( \Delta 2 \) in this problem, so slack is made up between FF and L1.

A similar situation exists with the loopback from L4 except that any slack borrowing must now be made up in two periods. Therefore:

\[ \Delta 1 + \Delta 2 + \Delta 3 + \Delta 4 \leq 2T_{eye}. \]

Depending on whether the longest delay occurs in \( \Delta 1 \) or \( \Delta 2 \) versus \( \Delta 3 \) or \( \Delta 4 \), either the first or second of these conclusions sets the minimum clock period.
a) Longest delay is $\Delta 2$, so this sets the clk delay:

$$T_{clk} \geq \Delta 1 + \Delta 2 \quad T_{clk} \geq 300 + 400$$

$$\boxed{T_{clk} \geq 700 \text{ ps}}$$

b) Longest delay is $\Delta 4$, so the second constraint sets the clock delay:

$$2T_{clk} \geq \Delta 1 + \Delta 2 + \Delta 3 + \Delta 4 \quad 2T_{clk} \geq 300 + 400 + 400 + 550$$

$$2T_{clk} \geq 1650 \text{ ps} \quad \boxed{T_{clk} = 825 \text{ ps}}$$

c) Longest delay is $\Delta 2$, so again the first condition sets the clock delay:

$$T_{clk} \geq \Delta 1 + \Delta 2 \quad T_{clk} \geq 300 + 900$$

$$\boxed{T_{clk} \geq 1200 \text{ ps}}$$
With skew:

To think about skew, I considered what would happen if you had skew on the first and second latch relative to the FF clock (I just considered the first two latches because these have the longest logic delay and due to the slack passing constraint discussed in par I, if the first two stages work, the next ones should also work.)

Try with a period of 700 ps.
- A diagram shows two signals labeled FF and L1, with a timeline indicating periods 350 to 1400 and 450 to 1500.
- Text annotations include "transmit" and "transmit output."
Looking at these four cases, we can see that regardless of how the loops skew is placed relative to FF→L₁ or FF→L₂, the skew is always made up during multiple cycles of looping. I think the only way this wouldn't be true is if one of the delay was less than 100ps because then a latch could potentially miss latching the data due to skew. Therefore, I concluded that latches in loopback like this are very resilient to skew and that as long as the skew is smaller than the delay, it should not affect the minimum necessary clock delay because the skew is made up over cycles so you don't get infinite slack passing where the latches would miss data. Therefore the minimum necessary clock periods with skew are the same.

a) 400ps
b) 825ps
(c) 1200ps

minimum necessary clock with skew.
a) Here is a somewhat simplified version of this circuit:

First, these structures are pulse generators that have the width of tiny and detect either a rising edge (top) or falling edge (bottom).

The gate is a dynamic gate that is discharged if CLK, nCLK and either the rising or falling edge pulse generators are high.

Since nCLK is an inverted, delayed version of CLK, if you AND CLK and nCLK (like this gate does), you get a signal that holds high shortly after the CLK rising edge.
Essentially, this dynamic gate discharges if the data transitions (rising or falling edge) around the rising edge of the clock. The exact details of how this corresponds to setup and hold time are discussed in part b, but essentially if transitions happen to close to the rising edge of the flip flop, an error signal is raised. Unlike other razor FF that actually compare the value stored in the flip flop and a shadow latch, this design just indicates that data has violated the setup or hold time. This might be used to correct a microarchitectural error by determining an error has likely occurred and then possibly re-executing that instruction. (Rather than actually correcting a particular piece of data as you might do with a lower level razor FF.)

b) To determine the timing window, let’s consider what happens when clk + nclk is high (time during which an error can occur):

Note: we want the worst-case (smallest window) so we should consider the shorter delay of the rising edge detector when looking at the earliest pulse that can be detected. Similarly, we should consider the longer delay of the falling pulse detector for determining the latest transition that will be detected.
c) The min-delay constraint (or the hold time) is just the latest time the data can transition after the clock edge when the razor TF will detect an error. In part b, we found this was:

\[
\text{hold} = \text{delay} - t_{\text{gate}} - 2 \times \text{inv} - \text{twnap}
\]

This is the worst (shortest hold time detectable) for a falling edge.

d) The first version of latches we considered was the shadow latch, which looked like:

![Diagram of shadow latch]

The basic idea of this razor FF is that data gets captured by both the FF and shadow latch. The shadow latch is configured so that in the worst case voltage scaling it will never have a setup error. The latch and FF outputs are compared and if they’re different, an error is detected and the (correct) bit from the latch is put in the data chain at the next available time, and the inst. one paused for a cycle to feed in the corrected data.
To find the timing window, we want to find how close around a rising clock edge a data change will be detected. Also, we want the worst case (smallest window). First, consider how much before the rising edge a transition is detected. i.e. the pulse discharges the dynamic gate and keeps an error.

The earliest we can see a pulse is when \( \text{CLK} = \text{CLK} \) is high AND after the gate delay. For a rising edge, the edge happens \( t_{\text{out}} \) time before the pulse. If we consider the rising edge of the clock to be at “zero”, then the start of the window happens at

\[
t_{\text{start}} = t_{\text{gate}} + t_{\text{in}} + t_{\text{trans}}
\]

The negative of this quantity is also the setup time.

The latest data transition we can detect is one where the pulse occurs just at the end of the negative pulse minus the gate delay. Such a pulse (or worst on a falling edge) happens \( 2t_{\text{in}} + t_{\text{trans}} \) before this condition. Therefore, again relative to the rising clk edge, the window ends at

\[
t_{\text{end}} = \text{delay} - t_{\text{gate}} - 2t_{\text{in}} - t_{\text{trans}}
\]

To find the window:

\[
\text{window} = t_{\text{end}} - t_{\text{start}}
\]

\[
= \text{delay} - t_{\text{gate}} - 2t_{\text{in}} - t_{\text{trans}} - (t_{\text{gate}} - t_{\text{in}} - t_{\text{trans}})
\]
Significantly, these types of "razor FFs" are often used for voltage scaling where the voltage is decreased until the "razor" detects errors. The problem is that as the voltage is scaled, the flip-flop can become metastable if its setup time constraints are changed due to scaling. This is often fixed with a metastability detector which compares the FF output at two times (a tiny apart and raises an error if they're different (the FF is probably metastable). This works pretty well, but if the FF output voltage is near the metastable voltage of the metastability detector, that can also be metastable.

This particular metastability issue isn't a problem for this flip-flop because the FF output is not actually compared to anything. Instead, this circuit is just looking to see if setup/hold errors might have occurred, but doesn't actually use the FF output to determine this. Therefore, even if the FF output is metastable, this design will not have metastability problems due to the FF.

However, this design is still not immune to metastability

Consider the following two situations:

![Diagram](attachment:image.png)
In either of the cases, the pulse begins to discharge the
dynamic gate, but because the pulse signal goes low before
the full dynamic transition, the output might not be fully
discharged. If this signal is in the metastable region of the
inverter that follows it, the error signal can actually be
meta-stable.