UNIVERSITY OF CALIFORNIA, BERKELEY
Department of Electrical Engineering and Computer Sciences
EE251B Advanced Digital Circuits and Systems

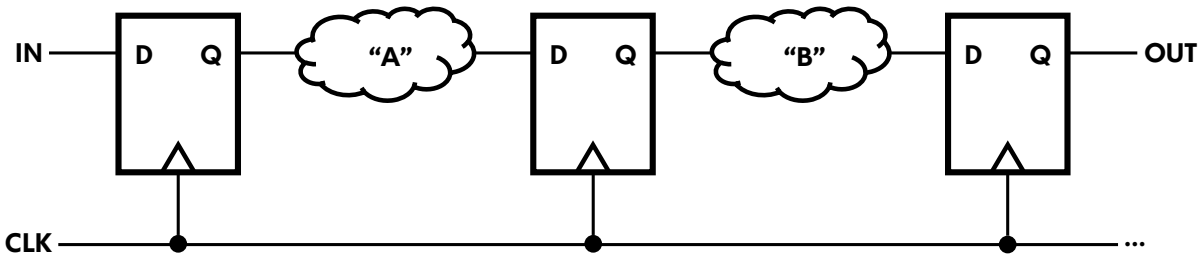
Spring 2024, Prof. Borivoje Nikolic
Homework 3 Solutions

Issued: Thursday March 7, 2024
Due: Friday March 22, 2024

---

1 Time Borrowing (40 %)

The following circuit has been used to pipeline the calculation of combinational logic block `A` followed by combinational logic block `B`, but the propagation delays of `A` and `B` are mismatched and other considerations in the system make impossible to rebalance them to equalize their delays. This problem considers a different approach to improving the overall timing. To make the calculations less tedious, assume that all flip-flops and latches have zero setup and hold time, 100 ps clock to output propagation delay, and there is no clock skew within the circuit.



(a) Suppose $t_{\text{pd,A}} = 2\,\text{ns}$ and $t_{\text{pd,B}} = 1\,\text{ns}$. What is the maximum clock frequency f_{clk} that the circuit can support without setup or hold time violations?

The minimum clock period is limited by the slower logic block, `A`:

$$T_{\text{clk}} \geq t_{\text{clk-q,ff}} + t_{\text{pd,A}} = 2.1\,\text{ns} \tag{1}$$
$$f_{\text{clk}} \leq 476.2\,\text{MHz} \tag{2}$$

On the second cycle, the computation of `B` finishes early.

(b) If the middle flip-flop is swapped for a positive latch, what is the new maximum clock frequency? Explain how the latch creates this change in performance.

It takes 3 clock cycles for the data to propagate from `IN` to `OUT`. Swapping for a positive latch in the middle stage allows the computation of `A` to finish as late as 1.5 cycles and still be captured properly, whereas before it had to finish at 1 cycle to be captured by the middle flip-flop:

$$1.5 \times T_{\text{clk}} \geq t_{\text{clk-q,ff}} + t_{\text{pd,A}} = 2.1\,\text{ns} \tag{3}$$
$$T_{\text{clk}} \geq 1.4\,\text{ns} \tag{4}$$

But, we also need `B` to finish before the third clock edge in the time remaining after `A` completes. Therefore, T_{clk} must also satisfy a second inequality in terms of both propagation delays:

$$2 \times T_{\text{clk}} \geq t_{\text{clk-q,ff}} + t_{\text{pd,A}} + t_{\text{pd,B}} = 3.1\,\text{ns} \tag{5}$$

$$T_{\text{clk}} \geq 1.55\,\text{ns} \tag{6}$$

We are limited by the more restrictive of the two inequalities (the second), yielding:

$$f_{\text{clk}} \leq 645.2\,\text{MHz} \tag{7}$$

Note that for simplicity this assumes instantaneous $t_{\text{d-q,lat}}$ in the latch (since none was given – oops!). Slightly lower f_{clk} answers that assumed $t_{\text{d-q,lat}} = t_{\text{clk-q,ff}}$ are also accepted.

(c) Suppose that instead $t_{\text{pd,A}} = 1\,\text{ns}$ and $t_{\text{pd,B}} = 2\,\text{ns}$. What is the maximum clock frequency in the original flip-flop based circuit? If the middle flip-flop is swapped for a negative latch, what is the new maximum clock frequency? Explain how the latch creates this change in performance.

The original flip-flop based circuit is unaffected by swapping the propagation delays, the slower delay limits the maximum clock frequency either way. With the swap and a negative latch inserted, we can write two new inequalities that are identical to part (b), which produces the same result:

$$f_{\text{clk}} \leq 645.2\,\text{MHz} \tag{8}$$

(d) The clock distribution scheme in the chip creates a slight duty cycle distortion in `CLK` from mismatched low-high and high-low propagation delays in the clock buffers. The net effect is 100 ps extra time added to the on duration of the clock and 100 ps time substracted from the off duration of the clock. For example, at $f_{\text{clk}} = 1\,\text{GHz}$ the clock waveform would have a 60% duty cycle at the location of this circuit on the chip. Recalculate the maximum clock frequency for each case above (flip-flop based, positive latch inserted with $t_{\text{pd,A}} > t_{\text{pd,B}}$, and negative latch inserted with $t_{\text{pd,A}} < t_{\text{pd,B}}$). For each solution, explain why the duty cycle distortion affected the result in the way that it did.

i. **Flip-flop case:** The duty cycle distortion does not affect the time between consecutive pairs of rising edges, so the flip-flop circuit is unaffected and we have the same result as part (a):

$$f_{\text{clk}} \leq 476.2\,\text{MHz} \tag{9}$$

ii. **Positive latch case:** The increased on-duration of the clock adds to the amount of time we can borrow by 100 ps, which improves the T_{clk} constraint from the first inequality:

$$1.5 \times T_{\text{clk}} + 100\,\text{ps} \geq t_{\text{clk-q,ff}} + t_{\text{pd,A}} = 2.1\,\text{ns} \tag{10}$$

$$T_{\text{clk}} \geq 1.33\,\text{ns} \tag{11}$$

The second inequality is unaffected, since it depends on the time between consecutive pairs of rising edges:

$$2 \times T_{\text{clk}} \geq t_{\text{clk-q,ff}} + t_{\text{pd,A}} + t_{\text{pd,B}} = 3.1\,\text{ns} \tag{12}$$

$$T_{\text{clk}} \geq 1.55\,\text{ns} \tag{13}$$

Therefore the result with duty cycle distortion is still the same:

$$f_{\text{clk}} \leq 645.2\,\text{MHz} \tag{14}$$

iii. **Negative latch case:** In this case the duty cycle distortion reduces the amount of time we can borrow by 100 ps, which worsens the first constraint:

$$1.5 \times T_{\text{clk}} - 100\,\text{ps} \geq t_{\text{clk-q,ff}} + t_{\text{pd,A}} = 2.1\,\text{ns} \tag{15}$$

$$T_{\text{clk}} \geq 1.47\,\text{ns} \tag{16}$$

However, like in the positive latch case, we are still limited by the computation of the entire path:

$$2 \times T_{\text{clk}} \geq t_{\text{clk-q,ff}} + t_{\text{pd,A}} + t_{\text{pd,B}} = 3.1\,\text{ns} \tag{17}$$

$$T_{\text{clk}} \geq 1.55\,\text{ns} \tag{18}$$

Therefore the result with duty cycle distortion is still the same:

$$f_{\text{clk}} \leq 645.2\,\text{MHz} \tag{19}$$

(e) In parts (b) and (c), suppose we made the opposite choices of latch polarity (swap for a negative latch when $t_{\text{pd,A}} > t_{\text{pd,B}}$ and a positive latch when $t_{\text{pd,A}} < t_{\text{pd,B}}$). What would be the new maximum clock frequency in each case?

If we pick the opposite type of latch in (b) and (c), we can no longer borrow/pass slack in the direction required to improve the timing, so the result is the same as the flip-flop case in (a):

$$f_{\text{clk}} \leq 476.2\,\text{MHz} \tag{20}$$

2 Clock Gating (20 %)

(a) In the clock gating scheme below, define the required relationship of the listed flip-flop, latch, buffer delay, and clock cycle T_{clk} in order to avoid the glitching on the `Gated_Clk` signal.

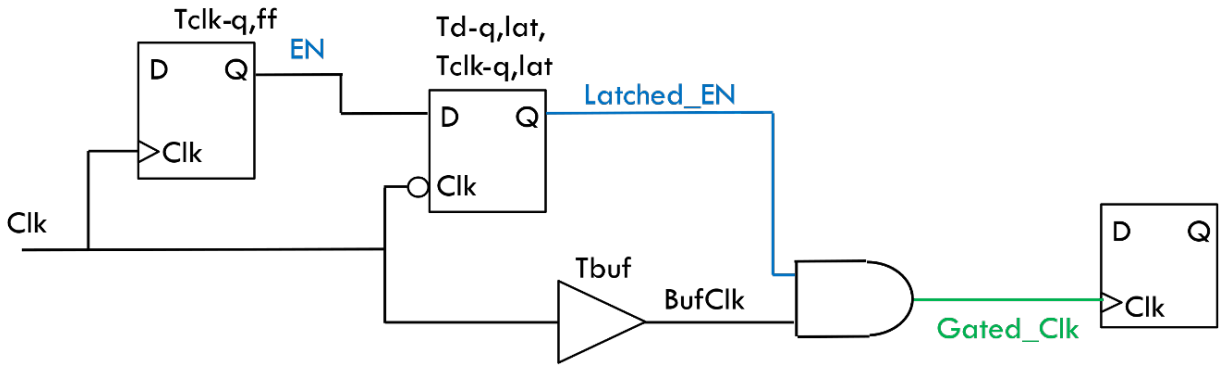
Both of the following conditions must be met:

$$T_{\text{buf}} < T_{\text{clk-q,lat}} \tag{21}$$

$$T_{\text{clk-q,ff}} < 0.5 \times T_{\text{clk}} \tag{22}$$

(b) Draw a timing diagram for a case that violates this relationship, causing glitching in `Gated_Clk`.

Any diagram showing either of the above being violated is accepted.

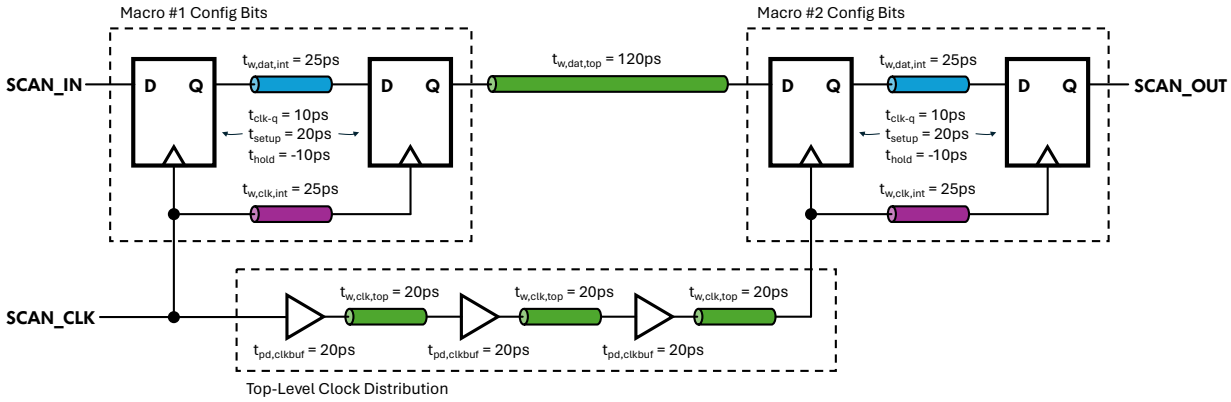


3 Chip Testing Data Interface (40 %)

This problem deals with the design of shift register based data interfaces for chip testing. Pay attention because it might help your chip tapeout someday!

It is almost always necessary to read and write test and configuration data to a new prototype chip, even an analog one. One simple way to do this is through a long shift register spanning all of the digital and analog/mixed-signal macros on the chip. This is often referred to as a scan register or scan chain. Each macro on the chip has a local scan register with timing verified during its place-and-route step. Then, at the top-level place-and-route step the scan registers are all wired in series and the scan clock is distributed to each macro. The scan interface doesn't necessarily need to be as fast as possible, but its design needs to be extremely robust against hold time violations because they will brick your ability to test your new chip. You can always slow the clock or boost the voltage if you have a setup issue, but as you discovered in the labs you can't count on repairing hold time violations this way.

(a) Let's restrict our discussion to scan write registers only, leaving reading data back from the chip as a future exercise for you to do on your own if needed. A simple way to implement a configuration scan chain is to put a flip-flop based shift register in each macro and wire all the inputs and outputs together at the top level, allowing the place-and-route tool to insert buffers as needed, like so:



Note that the tool inserted clock buffers but no data buffer because the clock rise/fall times were constrained tighter than the data rise/fall times. Let's suppose that there is a slight inaccuracy in the PDK parasitic extraction deck for the particular tool flow you are using (they are not always perfectly consistent, especially if we are dealing with a new process or open-source tools) and the *RC* delay through the particular metal layer used for top-level routing (the green-colored wire delays) is 10% faster than expected. How much setup and hold margin did the circuit originally have, and how much does it have now? Assume $f_{\text{clk}} = 100\,\text{MHz}$ for this scan chain interface because that's a typical maximum operating frequency that single-ended FPGA and CMOS IO cells can support (and you want to push this whole thing through digital tools with standard foundry-provided IP, avoiding the need to design custom differential IO cells and high-speed serial links).

Before wire delay error:

- Within the macro config bits there is 20 ps of hold time margin, since the data and clock delays are the same so $t_{\text{marg,hold}} = t_{\text{clk-q}} - t_{\text{hold}} = 20\,\text{ps}$ (note that the hold time

is negative, meaning that data can change 10 ps before clock edge arrives). The setup time margin in the config bits is given by $t_{\text{marg,setup}} = T_{\text{clk}} - t_{\text{setup}} - t_{\text{clk-q}} = 10,000\,\text{ps} - 20\,\text{ps} - 10\,\text{ps} = 9,970\,\text{ps} = 9.970\,\text{ns}$.

- Between the macros we need to carefully account for the clock skew caused by different clock path delays to the sending and receiving flip-flops. The clock distribution delay to the sending flip-flop is $t_{\text{pd,clkdist,tx}} = t_{\text{w,clk,int}} = 25\,\text{ps}$ and the delay to the receiving flip-flop (buffers and green wire delays) is $t_{\text{pd,clkdist,rx}} = 3 \times (t_{\text{pd,clkbuf}} + t_{\text{w,clk,top}}) = 120\,\text{ps}$, creating a positive skew of $t_{\text{skew}} = t_{\text{pd,clkdist,rx}} - t_{\text{pd,clkdist,tx}} = 95\,\text{ps}$.
  The setup time margin is given by:

$$\begin{aligned} t_{\text{marg,setup}} &= T_{\text{clk}} - t_{\text{clk-q}} - t_{\text{w,dat,top}} + t_{\text{skew}} - t_{\text{setup}} \\ &= 10,000\,\text{ps} - 10\,\text{ps} - 120\,\text{ps} + 95\,\text{ps} - 20\,\text{ps} \\ &= 9.945\,\text{ns} \end{aligned} \tag{23}$$

  The hold time margin is given by:

$$\begin{aligned} t_{\text{marg,hold}} &= t_{\text{clk-q}} + t_{\text{w,dat,top}} - t_{\text{skew}} - t_{\text{hold}} \\ &= 10\,\text{ps} + 120\,\text{ps} - 95\,\text{ps} + 10\,\text{ps} \\ &= 45\,\text{ns} \end{aligned} \tag{24}$$

  Note that the clock skew has opposite effects on the setup and hold margin – this will be useful for part (c).

Summary table (worst-case limiting values in bold):

	Intra-Macro	Inter-Macro (Top-Level)
Setup Slack	9.970 ns	**9.945 ns**
Hold Slack	**20 ps**	45 ps

After wire delay error:

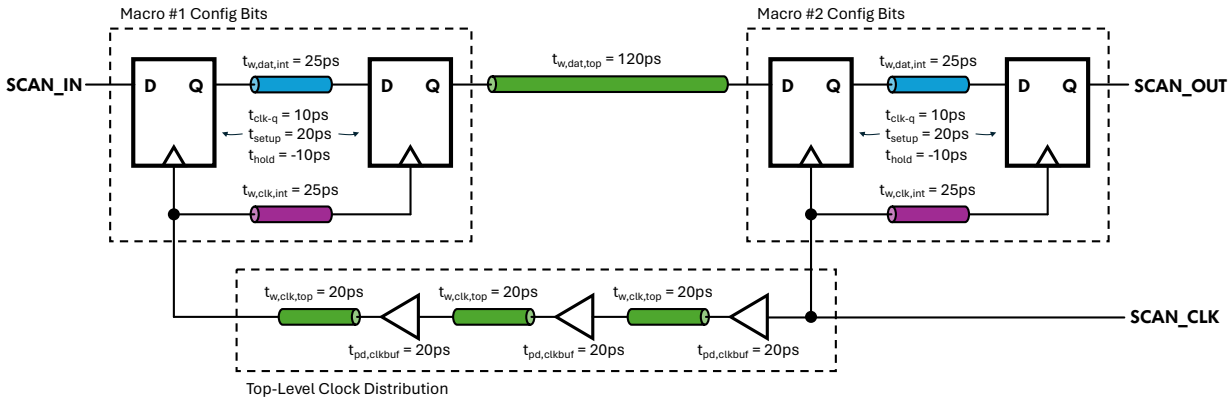
- There are no green wires within the macros affected by the *RC* delay error, so the macro-level timing margins stay the same.
- The green delays being 10% faster than expected causes the macro-macro data delay to decrease by 12 ps from 120 ps down to 108 ps, while the macro-macro clock distribution delay only decreases by 6 ps from 120 ps down to 114 ps. This results in a net reduction in the hold time margin of 6 ps and a net increase in the setup time margin by the same amount.

Summary table (worst-case limiting values in bold):

	Intra-Macro	Inter-Macro (Top-Level)
Setup Slack	9.97 ns	**9.951 ns**
Hold Slack	**20 ps**	39 ps

Although no timing violation occurs, we can observe that in principle, it is possible for a large enough discrepancy in the wire delay to create a hold time violation (the problem intended to illustrate this point, but the numbers were not selected properly for a hold time violation to occur).

(b) One trick to avoiding this type of glitch is described in the paper "Measurement of high-speed ADCs" by Lukas Kull and Danny Luu from IBM. Instead of allowing the top-level place-and-route tool to fully handle the clock distribution, force it to route the clock and data in opposite directions at the top level, clocking the last scan register first and the first scan register last, as shown below:



What are the new setup and hold margins, and by how much do the setup and hold margins change given the same error as before? Is there any percent error between expected and actual delays through the top-level (green-colored) routing paths that can create a hold time violation? If yes, what is it? If no, why not?

Before wire delay error:

The intra-macro timing margins are unaffected by the top-level clock distribution, but the clock skew for the inter-macro timing path now becomes:

$$\begin{aligned} t_{\text{skew}} &= -3 \times (t_{\text{pd,clkbuf}} + t_{\text{w,clk,top}}) - t_{\text{w,clk,int}} \\ &= -3 \times (20\,\text{ps} + 20\,\text{ps}) - 25\,\text{ps} \\ &= -145\,\text{ps} \end{aligned} \tag{25}$$

Therefore, the new inter-macro setup time margin is given by:

$$\begin{aligned} t_{\text{marg,setup}} &= T_{\text{clk}} - t_{\text{clk-q}} - t_{\text{w,dat,top}} + t_{\text{skew}} - t_{\text{setup}} \\ &= 10,000\,\text{ps} - 10\,\text{ps} - 120\,\text{ps} - 145\,\text{ps} - 20\,\text{ps} \\ &= 9.705\,\text{ns} \end{aligned} \tag{26}$$

and the new inter-macro hold time margin is given by:

$$\begin{aligned} t_{\text{marg,hold}} &= t_{\text{clk-q}} + t_{\text{w,dat,top}} - t_{\text{skew}} - t_{\text{hold}} \\ &= 10\,\text{ps} + 120\,\text{ps} + 145\,\text{ps} + 10\,\text{ps} \\ &= 285\,\text{ps} \end{aligned} \tag{27}$$

After wire delay error:

The intra-macro timing margins are again unaffected, but the clock skew for the inter-macro timing path now becomes:

$$\begin{aligned} t'_{\text{skew}} &= -3 \times (t_{\text{pd,clkbuf}} + 0.9 \times t_{\text{w,clk,top}}) - t_{\text{w,clk,int}} \\ &= -3 \times (20\,\text{ps} + 0.9 \times 20\,\text{ps}) - 25\,\text{ps} \\ &= -139\,\text{ps} \end{aligned} \tag{28}$$

The new inter-macro setup time margin is given by:

$$\begin{aligned} t'_{\text{marg,setup}} &= T_{\text{clk}} - t_{\text{clk-q}} - 0.9 \times t_{\text{w,dat,top}} + t'_{\text{skew}} - t_{\text{setup}} \\ &= 10,000\,\text{ps} - 10\,\text{ps} - 108\,\text{ps} - 139\,\text{ps} - 20\,\text{ps} \\ &= 9.723\,\text{ns} \end{aligned} \tag{29}$$

and the new inter-macro hold time margin is given by:

$$\begin{aligned} t'_{\text{marg,hold}} &= t_{\text{clk-q}} + 0.9 \times t_{\text{w,dat,top}} - t'_{\text{skew}} - t_{\text{hold}} \\ &= 10\,\text{ps} + 108\,\text{ps} + 139\,\text{ps} + 10\,\text{ps} \\ &= 267\,\text{ps} \end{aligned} \tag{30}$$

The setup margin increases by 18 ps while the hold margin decreases by 18 ps. Even if the green wire delays decreased by 100 %, we would still have positive hold time margin because the clock buffer delay is large enough to guarantee that the transmitting flip-flop is clocked later than the receiving flip-flop with enough margin to prevent a hold time violation. Note that routing the clock and data in opposite directions does not guarantee that there won't be a hold time violation by construction, we still need to check the specific numbers of the buffer delay and flip-flop timing parameters.

(c) In terms of setup and hold time margins at the chip top level, what are the advantages and disadvantages of routing the clock and signal in the same direction creating positive skew as in (a) vs. routing the clock and signal in opposite directions creating negative skew as in (b)? (*Hint*: We are looking for 4 answers to this part and we suggest writing them out in a 2×2 table.)

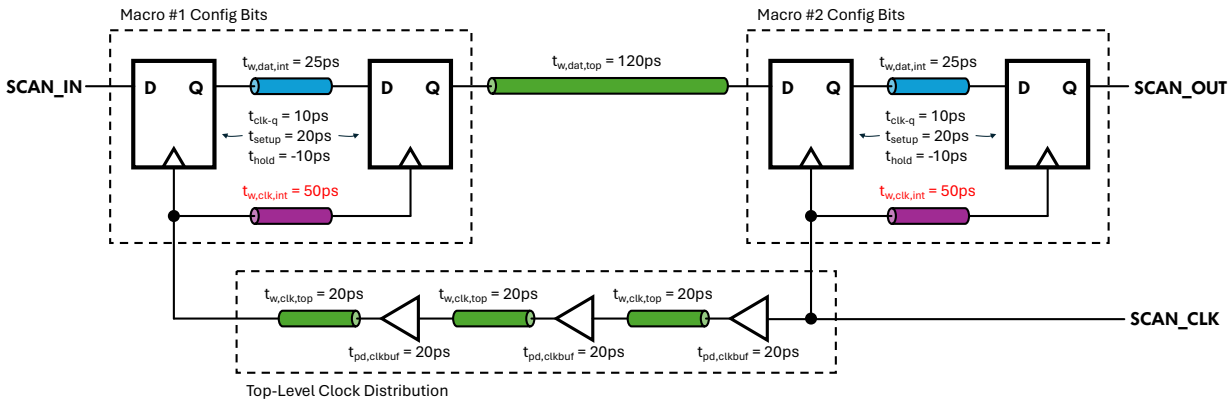
	Positive Clock Skew	Negative Clock Skew
Advantage	Increased setup margin, $\uparrow f_{\text{clk,max}}$	Increased hold time margin
Disadvantage	Reduced hold time margin	Reduced setup time margin, $\downarrow f_{\text{clk,max}}$

(d) Let's suppose we used this technique, but there was a bug in our macro-level timing constraints or extra wiring parasitics were added to some of the macro's internal wiring during the top-level assembly (for example, by placing a top-level power grid that the macro level place-and-route didn't know about over the wiring) and the fabricated chip ended up with the following timing parameters within a macro's scan register:

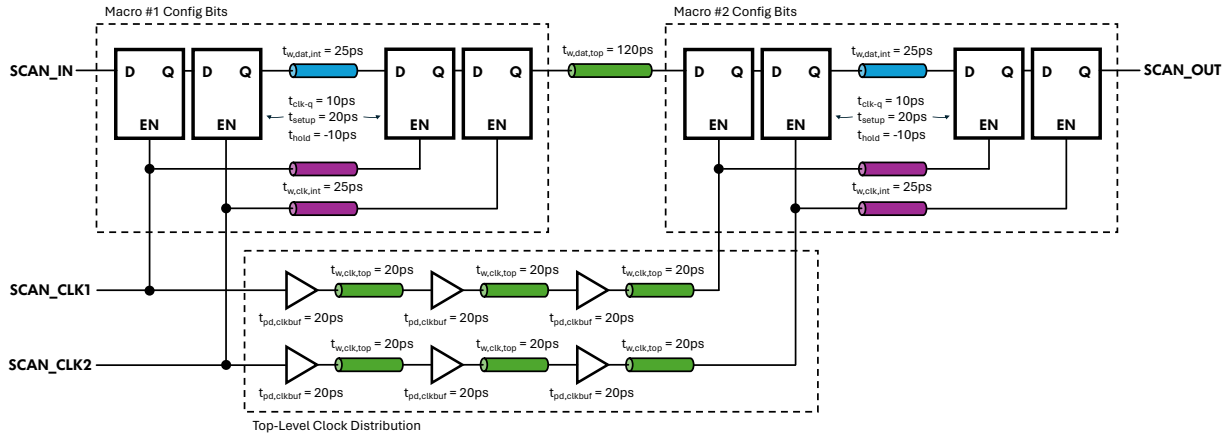
Where and how is timing violated? Is there any way to fix this post fabrication by adjusting the clock frequency or chip supply voltage?

The data from the first flip-flop arrives $t_{\text{clk-q}} + t_{\text{w,dat,int}} = 35\,\text{ps}$ after the rising edge of the each macro's local `SCAN_CLK` input, but the clock arrives with 50 ps delay to the second flip-flop, which has -10 ps hold time. This means the data at the input to the second flip-flop must be stable until 40 ps after the input `SCAN_CLK` edge, but it already changes 35 ps after the edge, producing a 5 ps hold time violation.

This hold time violation cannot be resolved post-fabrication, only setup time can reliably be fixed by adjusting the clock frequency. Adjusting the chip supply voltage affects both the hold time and the propagation delays and cannot generally be used to resolve hold time issues.



(e) Instead of using flip-flops, we can use a latch-based scan chain with two-phase non-overlapping clocks to guarantee setup and hold violation free operation regardless of any unexpected timing error in the scan chain, either at the top level or within a macro. Each data bit needs to pass through two latches instead of one flip-flop. The schematic of this circuit substituted into the design in part (a), where the place-and-route tool has free reign over the top-level clock distribution, is shown below:

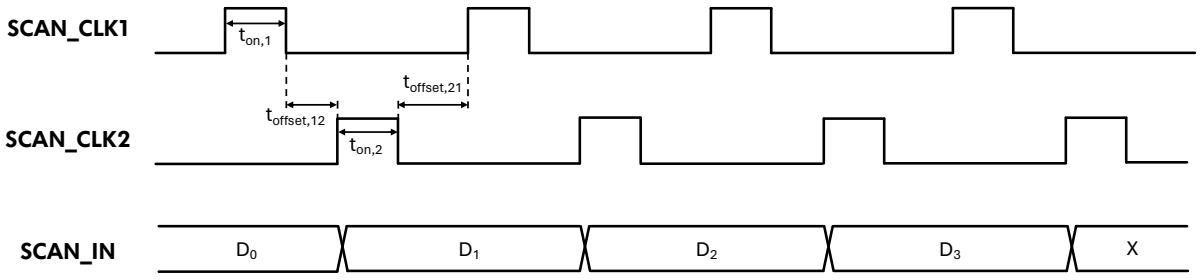


The following timing diagram defines the clock parameters that you can control externally:

Suppose that the 4 clock timing parameters are set up precisely such that the macro-macro timing arc considered in part (a) meets the hold time requirements with 5 ps margin. If the same 10% error as in part (a) occurs in this circuit, which of the 4 clock parameters would you need to change and by how much in order to fix the resulting timing violation?

In practice, we usually just set the 4 clock timing parameters to be equal, and if there's trouble we turn down the whole scan clock frequency until a scan loopback test works reliably. The fact that *any* timing violation possible in the circuit can be resolved this way is left as an exercise for the future in case you are curious.

If the green wires are 10% faster, then the clock arrives 6 ps sooner at the input latch of macro #2 but the data arrives 12 ps sooner, producing a 1 ps hold time violation if there was originally 5 ps hold time margin. Note that the transmitting latch is clocked from `SCAN_CLK_2` while the receiving latch is clocked from `SCAN_CLK_1`, and the hold time of the receiving latch



is specified with respect to the closing (falling) edge of `SCAN_CLK_1`. A hold time violation occurs when the data propagated by the next `SCAN_CLK_2` pulse arrives before the hold time of the receiving latch expires. Therefore, we can fix the hold time violation by making $t_{\text{offset,12}}$ more positive by $\geq 1\,\text{ps}$.

Unfortunately the numbers in this section were not selected properly, making the solution confusing because the only way for this hold time violation to occur is if we used overlapping clocks ($t_{\text{offset,12}} < 0$), but non-overlapping clocks are typically used with two-phase latch-based systems (as specified in the problem description) to avoid hold time violations. The violation could also occur if the latches had large enough positive hold times, but negative hold times are given. Turning the crank on the numbers without thinking through this carefully does produce the right answer without necessarily uncovering this inconsistency, but lenient partial credit is given since the formulation of the problem was incorrect.

In the original circuit, the hold time margin for the macro-macro timing path is given by:

$$\begin{aligned} t_{\text{marg,hold}} &= (t_{\text{w,clk,int}} + t_{\text{clk-q}} + t_{\text{w,dat,top}}) - (3t_{\text{pd,clkbuf}} + 3t_{\text{w,clk,top}}) - t_{\text{hold}} + t_{\text{offset,12}} \\ 5\,\text{ps} &= 25\,\text{ps} + 10\,\text{ps} + 120\,\text{ps} - 3\times 20\,\text{ps} - 3\times 20\,\text{ps} + 10\,\text{ps} + t_{\text{offset,12}} \\ -40\,\text{ps} &= t_{\text{offset,12}} \end{aligned} \tag{31}$$

So in the circuit with 10 % faster green wires, we would need to set $t_{\text{offset,12}} \geq -39\,\text{ps}$.

(f) In terms of chip area, power, and maximum scan register update rate, what are the advantages and disadvantages of flip-flop based scan chain and two-phase latch-based scan chain? In the two-phase latch-based case, assume that the $t_{\text{on,1}} = t_{\text{on,2}} = t_{\text{offset,12}} = t_{\text{offset,21}} \equiv t_{\text{pulse}}$ and the minimum pulse time corresponds to a half-period of the maximum f_{clk}, which is 100 MHz. Which of the two options would you prefer to use on your future chip tapeouts and why?

The latch-based scan chain consumes more chip area (double the number of latches vs. flip-flops, and flip-flops typically consume only about 50% more area), but operates at around half the maximum clock rate given the pulse duration limitation (100 MHz for flip-flop vs. 50 MHz for latch-based). The CV^2f dynamic power in the memory elements is similar between the two cases, since C almost doubles for typical latch vs. flip-flop designs while f halves. Note that the latch-based scan chain requires twice the amount of clock distribution operating at half the frequency, so dynamic power is similar but the area overhead is larger. No correct answer for which is better, it is subjective. Be aware of the risks/tradeoffs in your future chip designs!