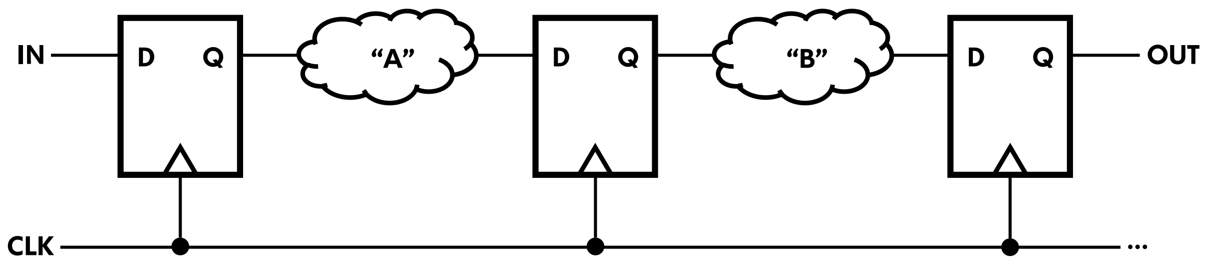
UNIVERSITY OF CALIFORNIA, BERKELEY
Department of Electrical Engineering and Computer Sciences

EE251B Advanced Digital Circuits and Systems

Spring 2024, Prof. Borivoje Nikolic — Issued: Thursday March 7, 2024
Homework 3 — 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?

(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.

(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.

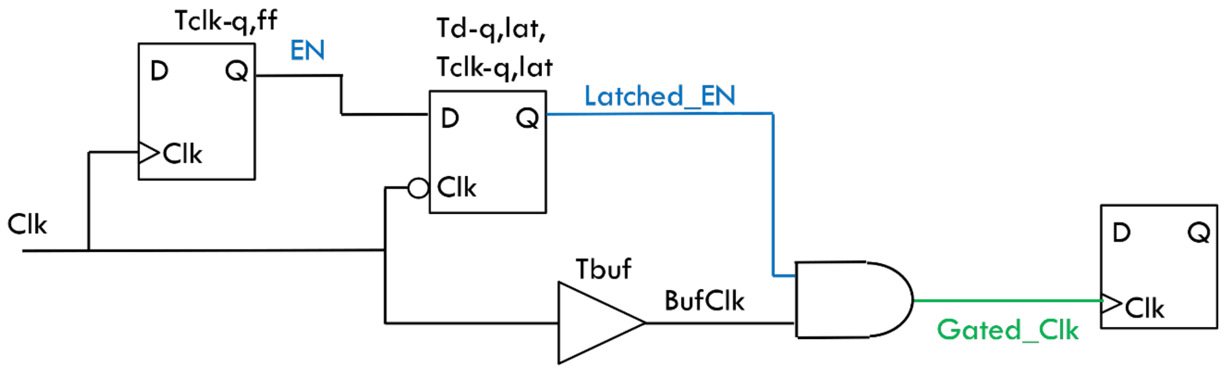
(d) The clock distribution scheme in the chip creates a slight duty cycle distortion in `CLK` from mismatched low-high and high-low propagation delays in the clock buffers. The net effect is 100 ps extra time added to the on duration of the clock and 100 ps time substracted from the off duration of the 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.

(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?

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.

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

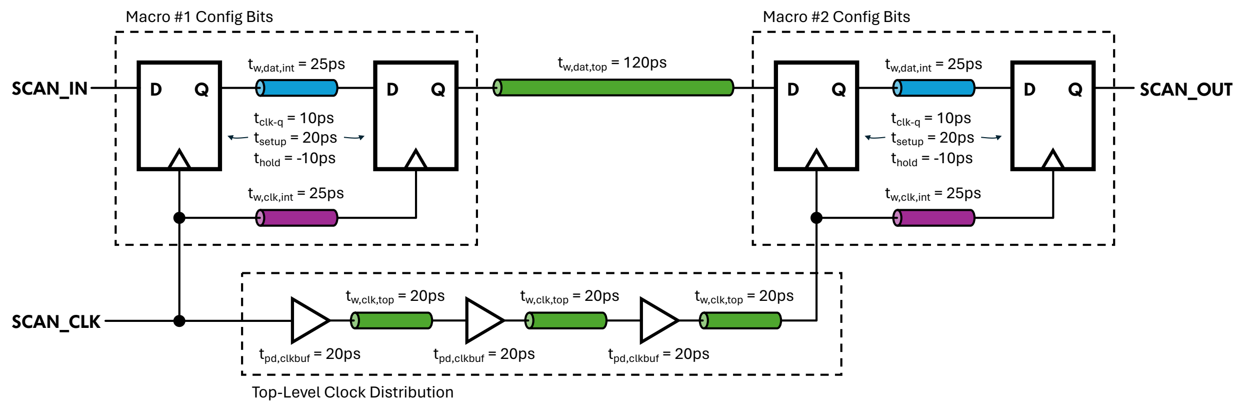


3 Chip Testing Data Interface (40 %)

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!

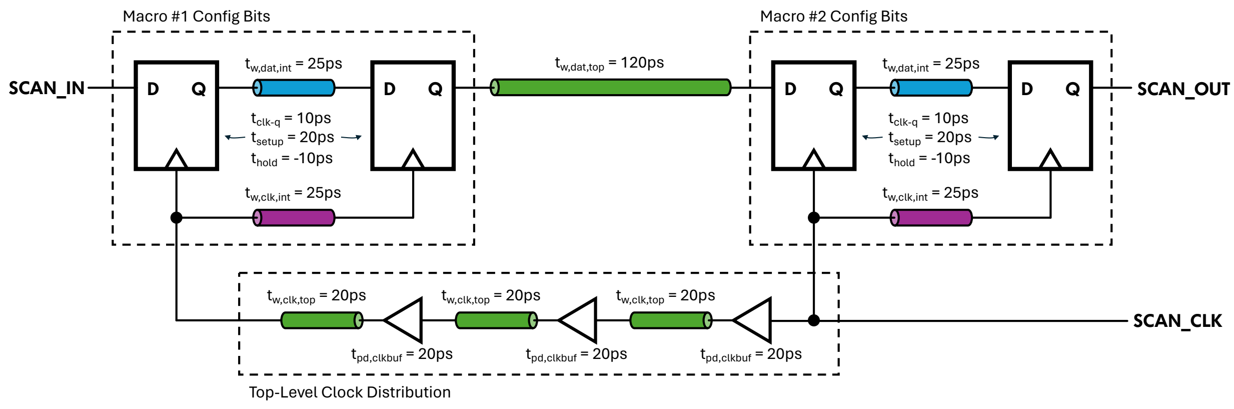
It is almost always necessary to read and write test and configuration data to a new prototype chip, even an analog one. One simple way to do this is through a long shift register spanning all of the digital and analog/mixed-signal macros on the chip. This is often referred to as a scan register or scan 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).

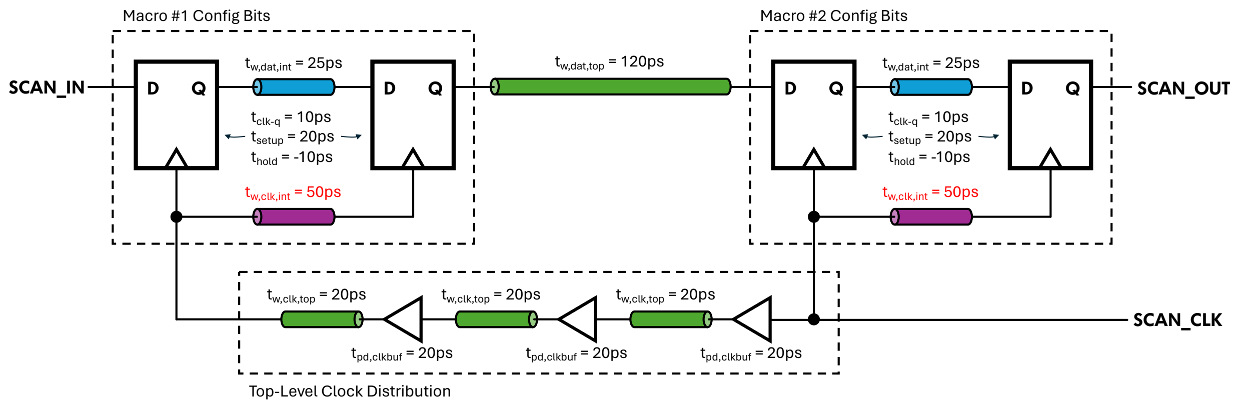
(b) One trick to avoiding this type of glitch is described in the paper "Measurement of high-speed ADCs" by Lukas Kull and Danny Luu from IBM. Instead of allowing the top-level place-and-route tool to fully handle the clock distribution, force it to route the clock and data in opposite directions at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

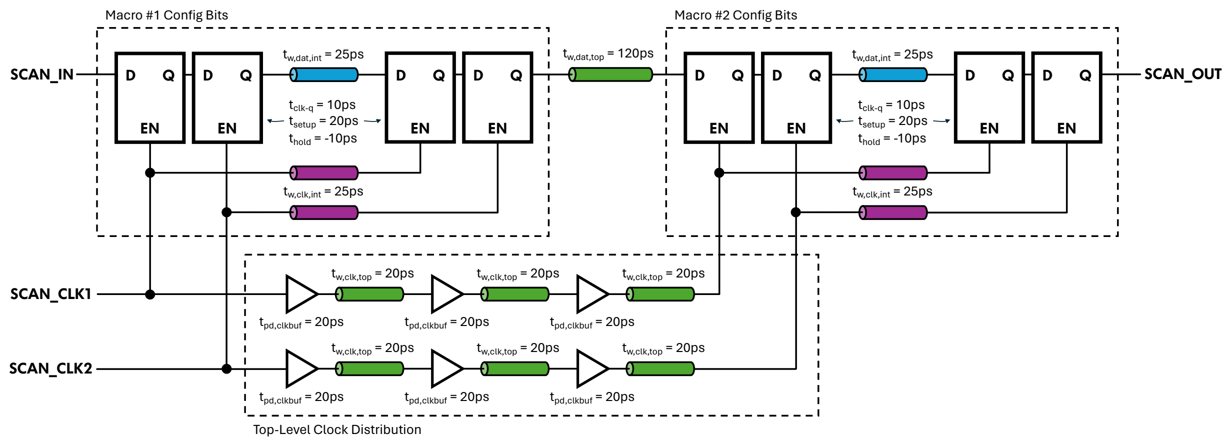
(c) In terms of setup and hold time margins at the chip top level, what are the advantages and disadvantages of routing the clock and signal in the same direction creating positive skew as in (a) vs. routing the clock and signal in opposite directions creating negative skew as in (b)? (*Hint*: We are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

(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:

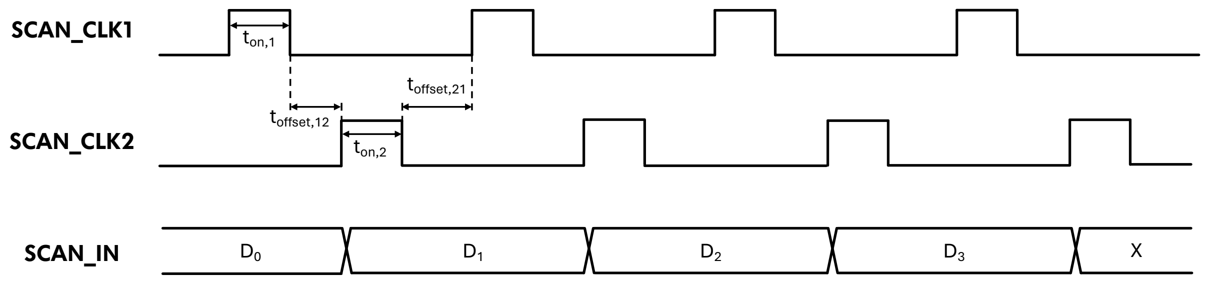


Where and how is timing violated? Is there any way to fix this post fabrication by adjusting the clock frequency or chip supply voltage?

(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.

(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?