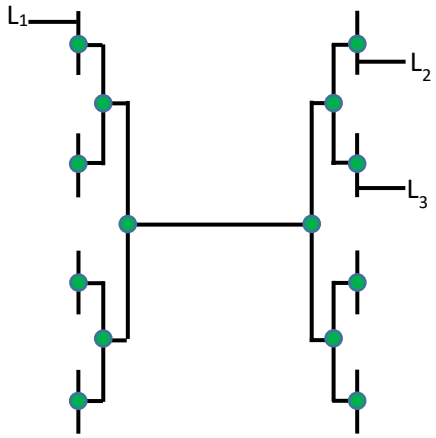
Homework #4 (clocking)

Problem #1

Consider a portion of a binary clock tree, as shown in the figure below. It comes from foil #32 from the lectures.



The green dots represent inverters. Loads L2 and L3 are identical; load L1 is larger. However, the final clock drivers are sized such that, at TT, the clock arrival time to all of the three loads is identical.

1. Which one(s) of process variation, OCV and di/dt noise will affect clock skew at the three loads? Jitter? Why?
2. Next, assume that we have active deskew, matching L1 to L2. Assume that the active deskew adjusts itself automatically during a chip reset, and then holds that delay setting. Which of the above issues will now be fixed?

Problem #2. Consider the circuit below. We have a pipeline, with two stages shown. We've only shown one flop in each pipe stage; in reality, each pipe stage would have many flops, and logic1 and logic2 would each have many inputs.

Assume that each of the NAND gates and inverters shown have delay=1, with jitter=±.1 and skew=±.2. Assume for simplicity that the flops each have $t_{setup}=t_{clk\rightarrow Q}=0$, and the transparent latch is the same and also has no propagation delay. Assume the both logic blocks logic1 and logic2 have delays of 20 to all outputs. Assume finally that the clock period is 35 time units.

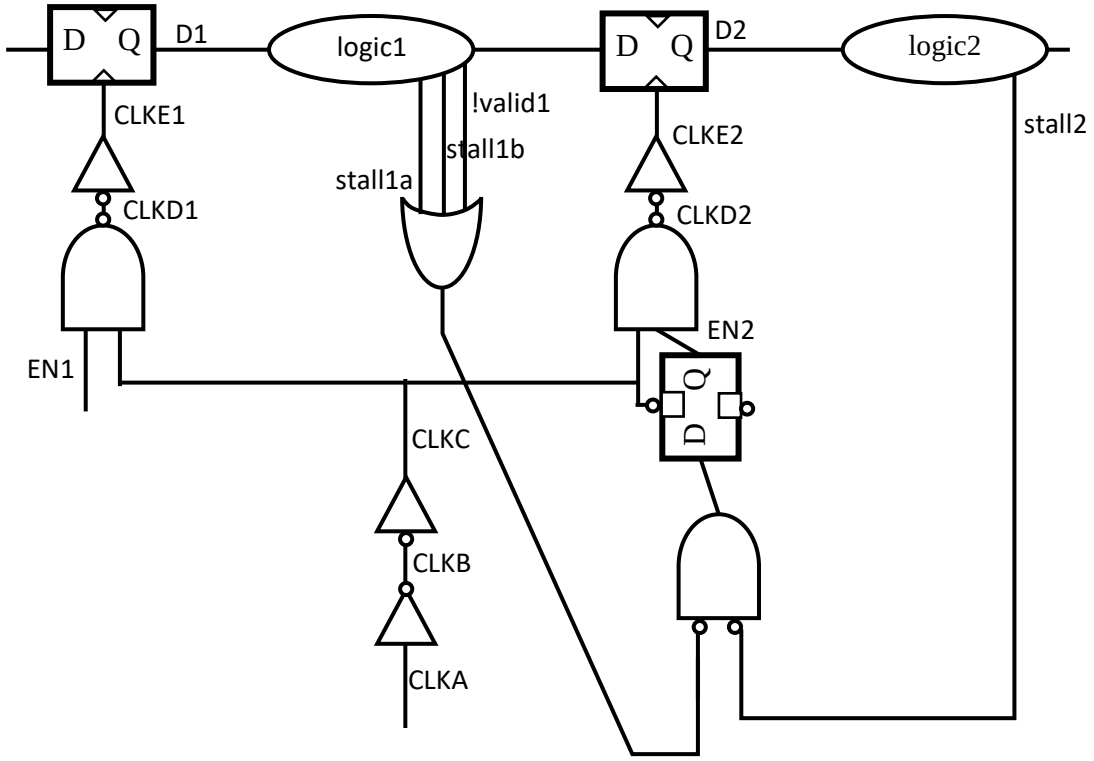
Consider the two critical paths to the clock enable EN2:

1. From CLKA through to CLKD1, to D1, to any of the logic1 outputs, and eventually to EN2
2. From CLKA through to CLKE2, to D2, to stall2, and eventually to EN2

How much delay slack does each of these two critical paths have? Remember that we've not accounted for wire delay; the more slack we have, the more distance wires can safely travel.

If we sacrificed some power and removed all of the logic that lets the first pipe stage clock-gate the second stage, how much more slack would we get?

We've said that there are two main reasons that clock enables tend to be critical paths. First, because they often travel long distances and are hence slow. Second, because they are terminated by early clocks and must thus finish 2-3 gates earlier than "normal" paths must finish. What does this say about how hard it is to do effective clock gating in a short-tick processor vs. a long-tick processor?



Problem #3

Now let's look at the control in more detail, and add a downstream pipe stage to make the problem more realistic. Consider three pipe stages (#1, #2 and #3). Each one has a valid signal (which comes straight from a flop, and indicates whether the pipe stage holds valid data) and a stall signal (which is high when the stage holds valid data and that data needs to stay in the current pipe stage and not advance).

1. Compute *validNext_2* (i.e., the input to the flop that drives the stage-2 valid bit), assuming that the valid bits are clocked unconditionally. You may use *valid_1*, *valid_2*, *valid_3*, *stalled_1*, *stalled_2* and *stalled_3* as inputs.
2. Compute *clkEn_2* (i.e., the expression that says pipe stage #2 will clock its data next cycle).
3. Re-compute the expressions for *validNext_2* and *clkEn_2* under the assumption that *valid_2* is not clocked unconditionally, but instead must use the same clock as pipe stage #2's data.
4. Given your results, what are the pros and cons of clocking the control unconditionally vs. reusing the datapath clock?

Problem #4

Assume that we have a mother PLL distributing a 100 MHz clock. We have two daughter PLLs, one multiplying by 9 to make a 900 MHz clock and the other by 15 to make 1.5 GHz. We want to transfer data between the two daughter domains.

To use a transfer technique such as a BGF, you must turn off 6 of the 15 clocks from the 1.5GHz domain so that both domains fire 9 times every 10000 ps. Pick six pulses to turn off so as to minimize the resulting clock skew, and calculate what the skew is for your choice. Note that the convention we chose in class was that the skew is positive when the receiver clock is later than the driver clock, and that the second part of this problem will assume a path from the 900 MHz domain to the 1.5 GHz domain, so please define the sign of the skew accordingly.

To simplify the problem, assume that (since 9 and 15 have the same ratio as 3 and 5), you can divide the 10000ps period into three 3333ps periods, and treat each one identically; i.e., turn off two clock pulses from the first five, and repeat that pattern three times.

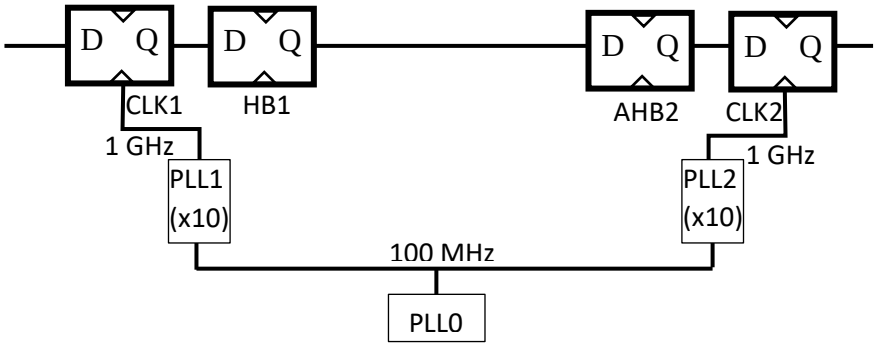
Also assume that you want to minimize the absolute value of the skew, and hence that a solution resulting in minimum skew of 0 and maximum of 100ps is worse than a solution resulting in skew between -20 and +80 ps.

Next, consider a flop-to-flop path using the above clocking scheme. Assume for simplicity that the flop setup time, hold time and clk-to-Q time are all zero. If we have a path from the 900 MHz domain to the 1.5 GHz domain, we want to pick the logic delay so that the setup and hold-time constraints have an equal amount of margin. How much delay would we have to add to make this happen?

Problem 5. Consider the heartbeat/anti-heartbeat crossing drawn below. Assume that the Mother PLL runs at 100 MHz, and that the two daughter PLLs each have a 10:1 clock ratio (so that the final clocks run at 1 GHz). Assume that the two flops have t_{setup}=t_{hold}=10ps. As we discussed in class, assume that HB2 and AHB2 are heartbeat-conditioned and anti-heartbeat-conditioned clocks similar to CLK1 and CLK2 respectively.

How much clock skew could we have between CLK1 and CLK2 and still have reliable operation? Measured in clock cycles at 1 GHz, how many clock cycles worth of skew is this?

Given how simple and how amazingly robust the heartbeat/anti-heartbeat crossing is, why would we ever want to use any other type of clock crossing circuit?



In class, we discussed using a FIFO to transfer data between clock domains. However, the skew and jitter between CLK1 and CLK2 can be quite large. As the skew and jitter get quite large, can we always build a FIFO that works reliably? Other than reliable operation, what problems occur when using a FIFO to overcome large clock skew and jitter?

Please turn in your assignment via the Provide cgi interface. You can use any reasonable format (including writing your answers by hand and taking a picture of the page with your phone).