Problem 1 – Activity Factor

Consider the decimal counter shown below. The static logic block is a 4-bit binary incrementor. Its input is a 4-bit binary number IN<3:0>, where IN<3> represents the most significant bit of the input number. Its output is a 4-bit binary number OUT<3:0> that has a numeric value 1 greater than, unless the input is 9, in which case the output is 0. I.e., if the input is 5 (0101), the output will be 6 (0110). If the input is 9 (1001), the output is (0000).

![Diagram of the circuit](image)

a) What are p₀ and p₁, the probabilities of a 0 and of a 1, for each of the four bits X<0> through X<3>? 

b) What are the activity factors α₀ through α₃ for each of the four bits X<0> through X<3>? 

c) If the capacitance at each node for X is 5 fF, the capacitance at each node for Y is 4 fF, and the circuit is clocked at 250 MHz, what is the dynamic power consumption of this circuit? Ignore all other capacitances, assume there is no glitching, and V_DD = 2.5 V.

d) In addition to the incrementor logic, you wish to add logic that detects when X<3:0> has a value 7. Recognizing that X<3:0> is never 15, this can be accomplished with a 3-input AND of bits <0> through <2>. If this is to be accomplished with only 2-input AND gates, which of the two choices below is preferable? Why?

![Design A](image)

![Design B](image)
Problem 2 – Flip Flops

After pulling an all nighter before a 10 am design review, you finish designing the flip-flop shown below. The key feature is that by using “tri-state” inverters, you can avoid using pass transistor switches. Tri-state inverters labeled 1-4 in the diagram act as normal inverters when their control input is high, but their outputs are high impedance (or open circuit, often referred to as logic “Z”) when their control input is low. A “!” in a signal name indicates logical compliment, or negative logic.

a) What type of memory is this (latch or register)?

b) Your lazy coworker gets into the office at 9:45 am, just before your design review, and while strolling by your desk comments “cute circuit, but it has a race condition problem.” What is the problem he is referring to?

c) At 9:55, with just five minutes left, you decide that you can alleviate the race condition by adding the inverter chain shown below to create several clock signals with edges that occur in a well defined sequence. Which clock signal would you use for each of the four tri-state inverters?

d) At the end of the review your boss is pleased with the design, but apologizes for not telling you at the beginning that you must also provide an asynchronous clear input. When the asynchronous clear input (CLR) is raised, the output should be set to 0 regardless of the clock signals, and should remain at 0 after the clear signal is removed (until the next time at 1 is stored). After this stressful morning, you head for an early lunch and then decide to take a nap to help you relax. Just as you are about to doze off, you realize that you can add the asynchronous clear function by adding a single transistor to the existing schematic. Show how this is done, and describe any sizing constraints on this transistor (no calculations – just explain it).
Problem 3 – Timing and Clock Skew

Consider the pipelined logic structure below, where signals advance from one latch to the next in each clock cycle. Assume latches L1 – L3 are all on the same clock and are positive transparent. The latches have a propagation delay given by $t_{latch} = 0.3$ ns when the clock is high. Combinational logic blocks A and B are static, and have logic delays that depend on their inputs. Minimum and maximum delays for each logic block are listed below.

- Block A: $t_{pmin} = 1.7$ ns; $t_{pmax} = 2.1$ ns
- Block B: $t_{pmin} = 1.2$ ns; $t_{pmax} = 1.6$ ns
- Block C: $t_{pmin} = 1.0$ ns; $t_{pmax} = 1.2$ ns (this is for all inputs)

![Diagram of pipelined logic structure]

a) Determine $t_{on,max}$, the maximum time that the clock pulse can be high (i.e. longest time the latches can be open for). Assume there is NO clock skew.

b) Determine the minimum clock period $t_{min}$ and the worst case latency $t_{pipeline}$ from the input of L1 to the output of C. Assume NO clock skew.

c) Now assume that the clock is routed from latch 1 (L1) to latch 3 (L3) in ascending order. Assume that the clock skew between subsequent latches is the same (i.e. the skew from L1 to L2 is the same as the skew from L2 to L3). Find the MINIMUM clock skew needed to safely run the clock with a 2.5 ns period.

d) What is the minimum clock period $t_{min}$ and worst case latency $t_{pipeline}$ from the input of L1 to the output of C if an additional latch is added between logic blocks B and C?

Bonus Problem – Wave Pipelining – *NOT GRADED!*

This problem will not be graded, but if you want to make sure you really understand pipeline timing, this should be a fun problem.

It’s another late night at the office, and you are struggling to meet throughput and latency requirements for the two stage pipelined logic path shown below. Your boss is insisting that the logic operate at 800 MHz so that you can beat the evil competitor’s expected product at 666 MHz, and this is the last logic block that doesn’t meet the timing specification. The registers have $t_{hold} = 0$ ns, $t_{c-q} = 0.3$ ns, and $t_{su} = 0.2$ ns. The minimum and maximum logic delays for logic blocks A and B are given below. These blocks have multiple inputs and outputs, so these delays mean that starting from the time the last input is setup, some outputs may be available as soon as
$t_{pmin}$, and others may take as long as $t_{pmax}$. This implies that the inputs to each block must be stable for at least $t_{pmax}$ to guarantee that a correct output is ever generated. Unfortunately, you are convinced that $t_{pmax}$ cannot be reduced for either logic block.

Block A: $t_{pmin} = 1.0$ ns, $t_{pmax} = 1.2$ ns  
Block B: $t_{pmin} = 0.6$ ns, $t_{pmax} = 0.7$ ns

Waiting for the second pot of coffee to finish brewing, you have a stroke of brilliance. You realize that the registers actually take up a lot of the timing budget. If you could get rid of the middle register, and operate the logic as shown below, you could save a little bit of time. In this case, the registers are clocked faster than the total delay through A and B, and there are two different groups of data flowing through the logic at any time. I.e., on one clock edge data is launched into logic block A. As that data flow progresses into logic block B, a new set of data is launched into block A. Two clocks after it is launched, the first set of data has reached the output of B and is latched, and the second set of data is progressing from A to B. This is referred to as “wave pipelining”, since multiple waves of data flow through a logic path without registers to separate them.

a) What is the minimum clock period you can achieve with this technique?

b) What is the maximum clock period that will operate correctly?

c) What is the MINIMUM value of $t_{pmin}$ for block A for this pipeline to operate correctly at 800 MHz?

d) What are some drawbacks/limits of the wave pipelining technique?  *Hints: think about previous questions, what if there were more logic blocks, etc?*