EECS 312: Digital Integrated Circuits

Homework #6

Reading: 7.3.2, 7.5.1, 7.6.1, 12.2.1

1. **Schmitt Triggers**: You are designing circuits for a product that is intended to operate in space, exposed to a significant amount of radiation-based noise. You must design a Schmitt trigger to minimize energy that will ensure that the input to the trigger can tolerate 40 fC of alpha-particle charge. The alpha particles deliver charge to the node, but do not remove existing charge. The input node has 18 fF of wiring capacitance and some additional capacitance from the gate of the Schmitt trigger (consider C_{gc} only).
   - **a.** Suppose you use the Schmitt trigger from Figure 7-47 (page 365) with the input inverter and feedback inverter having W_p = 2um and W_n = 1um. Size M4 and M3 of this trigger to minimize energy while providing sufficient noise rejection from alpha particles.
   - **b.** What alternative structure can provide noise rejection in this scenario? (Besides another alternative Schmitt trigger)

2. **Pipelining:** The Pentium II (running at 400 MHz) was originally fabricated in 250nm process technology and had a FO4 gate delay of 40 ps. Each edge-triggered register in the design had a setup time of 2 FO4 delays and a clock-q delay time of 2 FO4 delays.
   - **a.** Find the pipelining overhead of due to registers.
   - **b.** Now consider the Pentium 4, fabricated in 130nm technology at a clock frequency of 3GHz. Using constant-field scaling, find the FO4 delay and use this to calculate the pipelining overhead of the flip-flops (assuming the flip-flops have a fixed 2 FO4 setup and clock-q delay time).
   - **c.** Architects claim that the upper limit on the overhead of pipelining is ~25% (above this number performance suffers too greatly). Conventional CMOS is projected to reach fundamental barriers at the 15nm process technology. Considering these two limitations, use constant-field scaling to calculate the upper limit on clock frequency. Again consider setup and clk-Q delay to be 2 FO4 delays.

3. **C^2MOS Latch:** Consider the C^2MOS gate below in Figure 2.1.
   - **a.** Does this circuit (2.1) function the same as the conventional C^2MOS latch from the text (Figure 7-26)? Explain.
   - **b.** Does this new latch have any delay advantage or disadvantage in T_{CQ} and T_{DQ} over the previous C^2MOS design? Explain briefly. Which metric is more important in latch-based design? Justify your answer.
   - **c.** Can you think of an advantage and disadvantage of the new C^2MOS gate (2.1), unrelated to delay?
   - **d.** Pipeline the following function with C^2MOS latches, breaking the work into the smallest possible chunks. How many pipeline stages can we divide the function into with C^2MOS latches? Draw a simple diagram showing your pipeline. *Assume that all 1 and 2 input gates will have roughly equal propagation delay for this problem. Use parallelism during each cycle to minimize number of gates in the longest timing path, but divide the function into the smallest and fastest pipelined operations that you can design.*

$$X = ((B + D)AC + E(BD))F$$

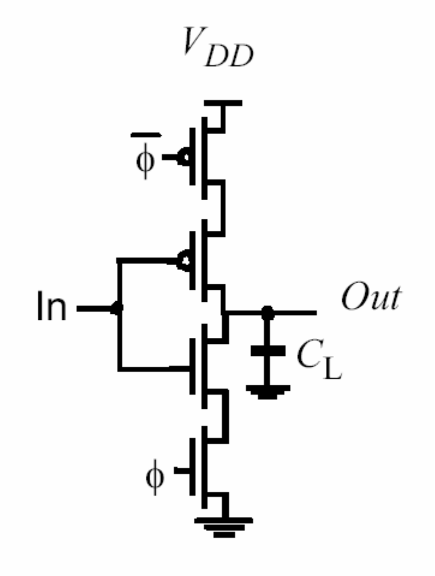

Figure 2.1

4. **ROM:** Consider a pseudo-NMOS based 32 X 32 ROM. Use C_{gc} (ignore overlap) and the 0.6 fF/um approximation for C_{db} to calculate relevant capacitances in the problem. Note that in the formulation we will ignore the significant wiring capacitances associated with memory elements. After wires are covered in class, we can explore their effects on memory access times. All NMOS elements in the ROMs are sized at W_n = 0.5 um. Assume that word lines are driven by static inverters with W_p = 2 um and W_n = 1um. Assume a step input to the word line drivers.

   a. Consider a 32 x 32 NOR ROM. Calculate the size of the pre-charge PMOS device if the V_{OL} must be 1 V.
   b. Calculate the worst-case access time of a 32 x 32 NOR ROM using the PMOS size from (a). Consider the worst-case programming. Measure from the input to the word line inverter to the bitline and consider the rise/fall time delay dependence with K=0.15.
   c. Sketch a 4 x 4 NOR ROM that stores 1010, 1110, 0010, and 0110.
   d. Sketch a 4 x 4 NAND ROM that stores 1010, 1110, 0010, and 0110.