For all problems, you can assume that all transistors have a channel length of 100nm and the following parameters (unless otherwise mentioned):

NMOS:

\[ V_{Tn} = 0.2V, \mu_n = 400 \text{ cm}^2/(V\cdot s), C_{ox} = 1.125 \mu\text{F/cm}^2, v_{sat} = 1e7 \text{ cm/s}, L=100\text{nm}, \gamma=\lambda=0 \]

PMOS:

\[ |V_{Tp}| = 0.2V, \mu_p = 200 \text{ cm}^2/(V\cdot s), C_{ox} = 1.125 \mu\text{F/cm}^2, v_{sat} = 1e7 \text{ cm/s}, L=100\text{nm}, \gamma=\lambda=0 \]

Problem 1: _____/ 10
Problem 2: _____/ 18
Problem 3: _____/ 15
Problem 4: _____/ 26
Problem 5: _____/ 21
Total: _____/ 90
PROBLEM 1: Logic Styles (10 points)

a) (5 pts) Assuming \( R_{s qp} = 2R_{sqn} \) and quadratic devices, what is the logical effort of the dynamic gate shown below from the A input?

\[
\text{Ref. inverter: } \quad \frac{1}{1.5} \quad \text{Drive fight between keeper and pull-down:}
\]

\[
R_{eff} = \frac{1}{R_{pd} - R_{keep}} = \frac{1}{1 - \frac{1}{1.5}} = 2
\]

\[
\text{LE} = \frac{R_{eff} \cdot C_{gate}}{R_{in} \cdot C_{in}} = \frac{2 \cdot 3}{1 \cdot 3} \quad \rightarrow \quad \boxed{\text{LEA} = 2}
\]

b) (5 pts) For the dual rail domino version of the same gate shown below, now what is the logical effort from the A input?

\* Because both \( \overline{\text{Out}} \) and \( \text{Out} \) are precharged high, during evaluation the cross-coupled PMOS "keepers" are off, and hence there is no drive fight.

\[
V_{gate} = 1 \quad , \quad C_{gate} = 3
\]

\[
\rightarrow \quad \boxed{\text{LEA} = 1}
\]
**PROBLEM 2: SRAM Design (18 pts)**

For this problem we will be looking at a 128x128 SRAM (i.e., each wordline drives 128 cells, and each bitline has 128 cells on it), with each cell shown below. The cell’s layout is 2μm tall and 2.5μm wide, and both the wordline and bitline wires are 0.1μm wide. You can assume that $C_G = C_D = 2fF/\mu m$, $R_{sqn} = 10k\Omega/\mu m$, $R_{sqp} = 20k\Omega/\mu m$, and that for the wordline and bitline wires, $R_w = 0.1\Omega/\mu m$ and $C_w = 0.2fF/\mu m$.

![SRAM Cell Diagram]

**a) (5 pts)** For this SRAM, what is the total capacitance on each wordline? What is the total capacitance on each bitline?

- **Wordline capacitance**:
  
  $$C_{WL} = N_{WL\text{cells}} \cdot (2W_{acc} \cdot C_w + W_w \cdot W_{cell})$$
  
  $$= 128 \cdot (0.48fF + 0.5fF)$$
  
  $$\Rightarrow C_{WL} = 125.44fF$$

- **Bitline capacitance**:
  
  $$C_{BL} = N_{BL\text{cells}} \cdot (W_{acc} \cdot C_d + W_w \cdot W_{cell})$$
  
  $$= 128 \cdot (0.24fF + 0.4fF)$$
  
  $$\Rightarrow C_{BL} = 81.92fF$$

**b) (4 pts)** Assuming this SRAM works at 1GHz and $V_{DD}=1.2V$, how much dynamic power is consumed due to the capacitance of the wordlines and bitlines?

- Only one wordline charged/discharged per cycle, but have 128 bitlines that are charged/discharged every cycle.

  $$C_{SW} = C_{WL} + 128 \times C_{BL} \approx 10.61pF$$

  $$P = C_{SW} V_{DD}^2 f_{cell} \Rightarrow P = 15.3mW$$
c) (9 pts) Now let’s see what happens when we change the size of the SRAM array. Assuming that the final gate driving the wordline is always an inverter sized to have a fanout of four, how many SRAM cells can you place on each wordline before the delay caused by the resistance of the wordline wire is equal to the delay caused by the resistance of the inverter that drives the wordline?

\[
C_{WL} = N_{cells} \cdot 0.98 \text{FF} \quad \rightarrow \quad C_{inv,driver} = N_{cells} \cdot \frac{0.98 \text{FF}}{4} = N_{cells} \cdot 0.245 \text{FF}
\]

\[
W_{NMOS,driver} = \frac{C_{inv,driver}}{C_{G}} \cdot \frac{1}{3} = 0.0408 \mu \text{m} \cdot N_{cells}
\]

\[
R_{driver} = \frac{L}{W_{NMOS}} \cdot R_{source} = \frac{24.49 \text{kΩ}}{N_{cells}}
\]

\[
R_{WL} = \frac{W_{cell}}{0.1 \mu \text{m}} \cdot R_{source} \cdot N_{cells} = 2.5 \Omega \cdot N_{cells}
\]

**RC Model:**

\[
\begin{array}{c}
\text{RC Model:} \\
\hline
\frac{1}{R_{driver}} \quad \frac{1}{C_{inv,driver}} \quad \frac{1}{R_{WL}} \\
\hline
\frac{1}{C_{WL/2}} \quad \frac{1}{C_{WL/2}}
\end{array}
\]

\[
\tau_{p,driver} = R_{driver} (C_{inv,driver} + C_{WL}) = 24.49 \text{kΩ} \cdot (0.245 \text{FF} + 0.98 \text{FF}) = 30 \text{ps} \quad (= 1 \tau_{F04})
\]

\[
\tau_{p,WL} = \frac{R_{WL} \cdot C_{WL}}{2} = 2.5 \Omega \cdot N_{cells} \cdot 0.98 \text{FF} \cdot N_{cells} = 1.225 \text{fs} \cdot N_{cells}^2
\]

\[
\tau_{p,WL} = \tau_{p,driver} \quad \rightarrow \quad N_{cells} = \frac{30 \text{ps}}{1.225 \text{fs}}
\]

\[
N_{cells} \approx 156.49 \\
\left( i.e., \approx 156 \text{ cells} \right)
\]
PROBLEM 3: Sequential Elements (15 points)
In this problem we will be examining the latch shown below.

![Latch Diagram]

a) (4 pts) Assuming the latch is ideal (i.e., has no delay, zero setup/hold time, etc.), fill in the waveforms for mid and Q given the clock and data inputs shown below.

![Waveforms]

clk
D
mid
Q
b) **(5 pts)** One of your fellow designers used this latch in their chip, and complains to you that their chip does not function correctly when they make the clock frequency too low. Could this latch design be the cause of the problem? If so, can you add some additional circuitry to the latch to fix this issue?

*Yes - the latch is dynamic when clk is low, so leakage can cause Q or mid to drift and eventually cause a failure.*

*Fix - add keepers:

\[ 
\begin{align*}
Q & \quad \text{or} \quad D \\
\text{clk} & \quad \text{mid} \\
\end{align*}
\]

\[ 
\begin{align*}
D & \quad \text{clk} \\
\text{mid} & \quad \text{Q} \\
\end{align*}
\]

c) **(6 pts)** Your fellow designer has also identified that on this same chip, the latch can fail when the load capacitance it drives is too small. Other than increasing the load capacitance, changing the sizes of the transistors, or applying any fixes that you may have made in part (b), is there any way you can modify the latch in order to eliminate this issue?

*Problem is charge sharing (Note what happens when clk=0, Q was previously VDD, and mid rises from 0 to VDD).*

*Fix - rearrange the transistors:

\[ 
\begin{align*}
D & \quad \text{clk} \\
\text{mid} & \quad \text{Q} \\
\end{align*}
\]
PROBLEM 4: Timing and Clock Distribution (26 points)

In this problem we will be examining the pipeline shown below. The minimum and maximum delays through the logic are annotated on the figure, and the flip-flops have the following properties: \( t_{\text{clk-q}} = 50\,\text{ps} \), \( t_{\text{setup}} = 50\,\text{ps} \), and \( t_{\text{hold}} = 50\,\text{ps} \). You can assume that the clock has no jitter.

![Pipeline Diagram]

a) (5 pts) Assuming there is no skew between clk₁, clk₂, and clk₃, what is the minimum clock cycle time for this pipeline? Are there any minimum delay (hold time) violations?

Largest CL delay: \( t_{\text{clk-q}} + t_{\text{p,\text{max}3}} + t_{\text{p,\text{max}3}} + t_{\text{setup}} \leq T_{\text{cyc}} \)

\[
T_{\text{cyc}} > 50\,\text{ps} + 250\,\text{ps} + 300\,\text{ps} + 50\,\text{ps}
\]

\[
T_{\text{cyc}} > 650\,\text{ps}
\]

Shortest CL delay: \( t_{\text{p,\text{min}2}} \)

\[
t_{\text{clk-q}} + t_{\text{p,\text{min}2}} > t_{\text{hold}}
\]

\[
50\,\text{ps} + 25\,\text{ps} > 50\,\text{ps}
\]

No hold time violation.
b) (5 pts) Now we’ll include the clock distribution network for this pipeline. Assuming that the delay of each inverter is nominally 50ps, but that each inverter’s delay varies randomly by +/-10%, now what is the minimum clock cycle time?

In worst case, path through CL₁ must handle an additional 1.1 × 2 × 50ps = 110ps of skew, while path through CL₂ and CL₃ only needs to handle (1.1 × 0.9) × 50ps = 49.5ps of skew.

Logically, the critical path becomes CL₁:

\[ t_{clk} + t_{p, max} + t_{setup} \leq T_{cyc} - t_{skew, 2} \]

\[ T_{cyc} > 50ps + 500ps + 50ps + 110ps \]

\[ T_{cyc} > 710ps \]
c) (6 pts) Under these same conditions (i.e., 50ps nominal inverter delay, +/-10% delay variation) this pipeline has a minimum delay (hold time) failure. Can you intentionally add delay into the circuit in order to fix this hold time violation without affecting the minimum clock cycle? Note that you can not modify the clock network – in other words, you can only add delay onto the logic paths. If this is possible, you should indicate where in the pipeline you want to add delay and calculate how much delay is necessary. If not, you should explain why it isn’t possible to eliminate the hold time violation without affecting the minimum cycle time.

\[ + \Delta \]

\[
\begin{array}{c}
\text{D} \\
\text{Q}
\end{array}
\]

\[
\begin{array}{c}
\text{CL}_1 \\
t_{\text{max}} = 500\text{ps} \\
t_{\text{min}} = 75\text{ps}
\end{array}
\]

\[
\begin{array}{c}
\text{D} \\
\text{Q}
\end{array}
\]

\[
\begin{array}{c}
\text{CL}_2 \\
t_{\text{max}} = 250\text{ps} \\
t_{\text{min}} = 25\text{ps}
\end{array}
\]

\[
\begin{array}{c}
\text{D} \\
\text{Q}
\end{array}
\]

\[
\begin{array}{c}
\text{CL}_3 \\
t_{\text{max}} = 300\text{ps} \\
t_{\text{min}} = 75\text{ps}
\end{array}
\]

\[
\begin{array}{c}
\text{D} \\
\text{Q}
\end{array}
\]

\[
\begin{array}{c}
\text{clk}_1 \\
\text{clk}_2 \\
\text{clk}
\end{array}
\]

Because of skew between clk_1 and clk_2, hold time problem occurs on path through CL_2:

\[ t_{\text{clk-1}} + t_{\text{p(min2)}} > t_{\text{hold}} + t_{\text{sk1,2}} \]

50ps + 25ps > 50ps + 110ps ×

Good news is we can just add delay on arc between CL_2 and flip driven by clk_1 (see above) w/o impacting worst-case delay through CL_1.

Let fix hold time:

\[ t_{\text{clk-1}} + t_{\text{p(min2)}} + t_d > t_{\text{hold}} + t_{\text{sk1,2}} \]

\[ t_d \geq 85\text{ps} \]
d) (10 pts) If we now replace the flip flops by pulsed latches with a pulse width of $t_w = 25\text{ps}$, the same $t_{\text{clk-q}}$, $t_{\text{setup}}$, and $t_{\text{hold}}$ as the flip-flops, and with $t_{\text{d-q}} = t_{\text{clk-q}} + t_{\text{setup}}$, what is the new minimum cycle time of the pipeline? Can this cycle time be achieved without failing any hold time constraints? Note that you still have the option to add delay like in part (c) in order to fix hold time violations.

*Since $CL_1$ is critical, best we can do is utilize the pulse to “borrow” some extra time for that path. In other words, will give an extra $25\text{ps}$ to $CL_1$, but this $25\text{ps}$ will have to be made up by path(s) through $CL_2$ & $CL_3$ (which in this case is fine). So:

$$t_{\text{clk-q}} + t_{\text{pmax}} + t_{\text{setup}} < T_{\text{cy}} - t_{\text{sk1,2}} + t_w$$

$T_{\text{cy}} > 685 \text{ps}$

*For hold time, we have to make sure that all paths (and in particular worst path) will still be OK. $CL_2$ is still worst min. path, so need to satisfy:

$$t_{\text{clk-q}} + t_{\text{pminz}} + t_d \geq t_{\text{hold}} + t_{\text{sk1,2}} + t_w$$

$$t_d \geq 110 \text{ps}$$
PROBLEM 5: Arithmetic (21 pts)

In this problem we will design a circuit that detects the location of the first “1” in a binary input signal. “First” is defined relative to the MSB, and all outputs from the circuit should be zero except for the position where the first “1” occurred. For example, if the input to the circuit was “00101011”, the output of the circuit should be “00100000”. You can assume that both the true and complement versions of the input are available.

a) (3 pts) Assuming the input signal is only 2 bits wide, draw a gate-level schematic (i.e., no transistors) showing how you would calculate the 2-bit output Out_{1:0}. Each “gate” can be an inverter, NAND2, or NOR2.

![Gate-level schematic for 2-bit output](image)

b) (6 pts) Draw a gate-level schematic indicating how you would implement this circuit for an 8-bit wide input using the minimum number of gates. Assuming that all of the gates have equal delays, how many gate delays are there on the critical path of this circuit?

![Gate-level schematic for 8-bit output](image)

Critical path is through chained NAND/NOR gates: $t_{\text{crit}} = 8$ gates
c) **(12 pts)** Your colleague Ace says that assuming that the delay of the gates is independent of their fanout, she can reduce the number of gate delays on the critical path of an 8-bit wide leading 1 detector to only 3. Draw a gate-level schematic of an implementation that achieves this delay.

(Hint: Think about how you can use your design from part (a) along with one additional block to implement something similar to a carry look-ahead adder. Don’t worry about the logical polarities – you can assume that you have both the true and complement versions of every intermediate signal in addition to the primary inputs.)

Conceptually, best we can do is to implement the “rippling” with a tree structure:

```
<table>
<thead>
<tr>
<th>In 7:6</th>
<th>1-del</th>
<th></th>
<th>2</th>
<th></th>
<th>Out 7:6</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>In 6:4</td>
<td>1-del</td>
<td></td>
<td>2</td>
<td></td>
<td>Out 6:4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>In 5:2</td>
<td>1-del</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>In 4:0</td>
<td>1-del</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

Same as part (a)

Critical path(s) indeed 3 gates (OR-OR-OR-K:11).