1. The increment computation shown in the lecture is performed using complementary NAND/NOR/INVERT circuitry (See Fig. 1). Use the same style of design (NAND, NOR, INVERT) to build a circuit that implements the Boolean equation $F = [AB] + [C + DE]$ Redesign your circuit using a compound gate. Show the compound gate transistor diagram.

Here is one example circuit:

![Compound Gate Diagram]

Compound Gate:

![Compound Gate Diagram](image-url)

$$\text{out} = [AB] + [C + DE]$$
2. The decrement computation shown in the lecture is a compound gate. Implement the following Boolean Equation as a compound gate:

\[ G = (A+C)(D+F)(A+E)(B + \overline{A}) \]

Redesign your circuit using complementary NOR/INVERT circuitry (no NANDs). Use truth tables to show that the computation is identical to the compound gate. Compare the number of transistors to the original design.

Note that the inverter is there to create positive logic. The compound gate outputs negative logic, \( \overline{G} \).
Number of transistors in the compound gate = 18

Number of transistor in the nor – nor implementation = 24

\[ G = [(A + C][D + F] [A + E)][B + /A] \]

<table>
<thead>
<tr>
<th>C</th>
<th>D</th>
<th>F</th>
<th>A</th>
<th>E</th>
<th>B</th>
<th>/A</th>
<th>output</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>x</td>
<td>x</td>
<td>0</td>
<td>x</td>
<td>x</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>x</td>
<td>0</td>
<td>0</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>0</td>
</tr>
<tr>
<td>x</td>
<td>x</td>
<td>x</td>
<td>0</td>
<td>0</td>
<td>x</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>x</td>
<td>x</td>
<td>x</td>
<td>1</td>
<td>x</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>x</td>
<td>1</td>
<td>x</td>
<td>1</td>
<td>x</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>x</td>
<td>1</td>
<td>x</td>
<td>1</td>
<td>1</td>
<td>x</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>x</td>
<td>1</td>
<td>x</td>
<td>1</td>
<td>1</td>
<td>x</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>x</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>x</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Note the use of don’t cares in the truth table.
3. Modify the register shown Figs. 1.13 b and 1.13 c so that the new value actually stored is the OR of the previous value stored and the new input value. Show any added logic.

Answer 3) Modification of the D FF

At the positive edge of the clock
Q = D or Q;

---

Diagram of the modified D FF circuit.
4. Design a multiplexer to select between one of seven inputs with transmission gates (like the two-to-one mux described in class). Show the transistor level schematic. Redesign the 7-to-1 mux using a compound gate followed by an inverter. Derive Boolean expressions from both circuits to insure that they both perform the same function. Hint: how many control signals do you need? That determines the characteristics of your circuit.

**Answer 4) 7: 1 MUX**

Using Pass transistor logic.

*Possible solution 1*
Output =
A/S0/S1/S2 + B/S0/S1/S2 + C/S0S1/S2 + D/S0S1/S2 + E/S0/S1S2 + F/S0/S1S2 + GS1S2
**Possible solution 2**

Output =
\[ A/S0/S1/S2 + B/S0/S1/S2 + C/S0/S1/S2 + D/S0/S1/S2 + E/S0/S1/S2 + F/S0/S1/S2 + G/S0/S1/S2 \]

**Compound Gate Implementation for solution 1**

Output =
Compound Gate implementation for solution 2
Output =
A/S0/S1/S2 + BS0/S1/S2 + C/S0S1/S2 + DS0/S1/S2 + E/S0/S1S2 + FS0/S1S2 + G/S0S1S2

A/S0/S1/S2 + BS0/S1/S2 + C/S0S1/S2 + DS0/S1/S2 + E/S0/S1S2 + FS0/S1S2 + GS1S2/S0