Given the organization shown below, solve the following four problems:

Problem#1- Show how the above 3-bit comparator works, and then explain what would be the time complexity of your algorithm, and the space complexity of your design when A and B are n bits each.

Solution:

\[ G_i = X_iY_i' \]
\[ E_i = X_iY_i + X_i'Y_i' \]
\[ S_i = X_i'Y_i \]

\[ G = G_3 + E3G2 + E3E2G1 + E3E2E1G0 \]
\[ E = E3E2E1E0 \]
\[ S = S3 + E3S2 + E3E2S1 + E3E2E1S0 \]

Time complexity is \( O(n) \) as for a n-bit comparator, it needs to be iterative.
Space complexity is \( O(n) \) as n 1-bit comparators are needed.
**Problem #2**- Show and explain why the above two MUXes are placed in the above design and how do they work and function specifically in here. Then in general, would you say that multiplexes are universal? Why? And then lastly show a tree implementation of a 16 input multiplexer using a set of 4-input multiplexers.

Solution:
For 1-bit MUX
\[ Y = S \cdot x_0 + S' \cdot x_1 \] (Gate level schematic is based on this expression)

When \( G \) is 1, \( A \) which is the larger number will be selected. When \( G \) is 0, \( B \) which is the larger number will be selected. Similarly, \( L \) can select the smaller one.

Mux is universal as mux can implement NOT and AND.

Suppose there are 4 bits selectors (S3,S2,S1,S0). S3 and S2 are the control bits for the first level of 4 muxes. The 16 inputs will go in to the 4 muxes in the first level. And the 4 outputs of the 4 muxes will go into the mux in the second level. S1 and S0 will be the selectors of the second level mux.

**Problem #3**- Show the logic circuit design of the above 3-bit serial adder subtractor, and then if this was to be extended to handle \( n \) bits, talk about the speed and hardware tradeoffs in replacing the serial adder with a look-ahead adder, and then with a sequential single Full Adder.

Solution:
3-bit serial adder subtractor

![3-bit serial adder subtractor diagram](image)

A more detailed figure will be
Time complexity for carry ripple is $O(n)$, for carry look-ahead adder is $O(\log n)$ and for single full adder is $O(n)$.

Space complexity for carry ripple is $O(n)$, for single full adder is $O(1)$.

**Problem #4-** Show the detailed design of the above counter using D flip-flops. Show how the counter can be implemented on a PSA module. Show how you can use a ROM to implement the combinational part of the counter.

For D flip-flops:
- $Z_2 = CNT'Z_2 + Z_2 Z_1' + CNT Z_1 Z_0 + S_2 CNT Z_2 Z_1 Z_0'$
- $Z_1 = CNT Z_1'Z_0+CNT'Z_1 + Z_2'Z_1Z_0' + S_1CNT Z_2 Z_1 Z_0'$
- $Z_0 = CNT' Z_0 + CNT Z_1' Z_0' + CNT Z_2' Z_0' + S_0 CNT Z_2 Z_1 Z_0'$

where $Z_2$, $Z_1$ and $Z_0$ are the present states stored in D flip-flop. $S_2$, $S_1$ and $S_0$ are from the adder subtractor which are the initial value of counter.

For PSA,
Based on the expression above, let the input sequence is CNT, $S_2$, $S_1$, $S_0$, $Z_2$, $Z_1$, $Z_0$, we have
- $H0V0$, $H0V9$
- $H1V9$, $H1V10$
- $H2V1$, $H2V11$, $H2V13$
- $H3V1$, $H3V5$, $H3V9$, $H3V11$, $H3V12$
- $H4V1$, $H4V10$, $H4V13$
- $H5V0$, $H5V11$
- $H6V8$, $H6V11$, $H6V12$
- $H7V0$, $H7V13$
- $H8V1$, $H8V9$, $H8V11$
- $H9V1$, $H9V7$, $H9V11$
- $H10V1$, $H10V6$, $H10V8$, $H10V10$, $H10V11$

and
- $H0Y0$, $H1Y0$, $H2Y0$, $H3Y0$
- $H4Y1$, $H5Y1$, $H6Y1$, $H7Y1$
- $H8Y2$, $H9Y2$, $H10Y2$, $H11Y2$
For ROM, based on the expression above

The schematic of counter

The content in ROM

address (CNT, S2, S1, S0, Z2, Z1, Z0), output (Z2, Z1, Z0)

<table>
<thead>
<tr>
<th>CNT</th>
<th>S2</th>
<th>S1</th>
<th>S0</th>
<th>Z2</th>
<th>Z1</th>
<th>Z0</th>
<th>Z2</th>
<th>Z1</th>
<th>Z0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000000</td>
<td></td>
<td></td>
<td></td>
<td>000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0000001</td>
<td></td>
<td></td>
<td></td>
<td>001</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0000111</td>
<td></td>
<td></td>
<td></td>
<td>111</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1000000</td>
<td></td>
<td></td>
<td></td>
<td>001</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1001000</td>
<td></td>
<td></td>
<td></td>
<td>001</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1111110</td>
<td></td>
<td></td>
<td></td>
<td>000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>