Problem #1  Module-11 counter

a) using a standard module-16 binary counter:

Outputs:

if LD=1 ⇒ Output = Input (load)
if clear=1 ⇒ Output = 0
if CNT = 1 ⇒ It counts
if up/down = 1 ⇒ It counts up.

b) Using a PSA

c) using FFs

<table>
<thead>
<tr>
<th>PS</th>
<th>NS</th>
<th>A'B'C'D'</th>
<th>AB'CD</th>
<th>$T_A$</th>
<th>$P_B$</th>
<th>$J_c$</th>
<th>$K_c$</th>
<th>$S_B$</th>
<th>$R_B$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0010</td>
<td>0010</td>
<td>0010</td>
<td>0010</td>
<td>0010</td>
<td>0010</td>
<td>0010</td>
<td>0010</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0110</td>
<td>0110</td>
<td>0110</td>
<td>0110</td>
<td>0110</td>
<td>0110</td>
<td>0110</td>
<td>0110</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1010</td>
<td>1010</td>
<td>1010</td>
<td>1010</td>
<td>1010</td>
<td>1010</td>
<td>1010</td>
<td>1010</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0001</td>
<td>0001</td>
<td>0001</td>
<td>0001</td>
<td>0001</td>
<td>0001</td>
<td>0001</td>
<td>0001</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0011</td>
<td>0011</td>
<td>0011</td>
<td>0011</td>
<td>0011</td>
<td>0011</td>
<td>0011</td>
<td>0011</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0111</td>
<td>0111</td>
<td>0111</td>
<td>0111</td>
<td>0111</td>
<td>0111</td>
<td>0111</td>
<td>0111</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1011</td>
<td>1011</td>
<td>1011</td>
<td>1011</td>
<td>1011</td>
<td>1011</td>
<td>1011</td>
<td>1011</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1001</td>
<td>1001</td>
<td>1001</td>
<td>1001</td>
<td>1001</td>
<td>1001</td>
<td>1001</td>
<td>1001</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1101</td>
<td>1101</td>
<td>1101</td>
<td>1101</td>
<td>1101</td>
<td>1101</td>
<td>1101</td>
<td>1101</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1111</td>
<td>1111</td>
<td>1111</td>
<td>1111</td>
<td>1111</td>
<td>1111</td>
<td>1111</td>
<td>1111</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Using excitation tables for $T$, $Jk$, $Rs$ FlipFlops, we find $T_A$, $JcKe$, $SdRd$.

and $D_B = B^*$

$A^* = \overline{A'B'C'D'} + \overline{AB'CD} = \overline{A'B'C'D'} + \overline{AB'CD}$

$B^* = \overline{A'B'C'D'} + \overline{AB'CD} + \overline{AB'CD} = \overline{A'B'C'D'} + \overline{AB'CD} + \overline{AB'CD} + \overline{AB'CD}$

$C^* = \overline{ABC'D'} + \overline{A'BC'D'} + \overline{A'BC'D'} + \overline{A'BC'D'} = \overline{A'BC'D'} + \overline{A'BC'D'} + \overline{A'BC'D'}$
c) using FFs.

\[ T_A = BCD + AC \]
\[ D_B = BC + BD + \overline{BCD} \]

\[ J_c = D \]
\[ K_c = D + A \]

\[ S_D = \overline{CD} + \overline{AD} \]
\[ R_D = D \]

**Note:** Since \( S \) and \( R \) can't be "1" at the same time, if you consider a "-" as "1" for \( S \), you need to consider it as "0" for \( R \) and vice versa.
d) using a ROM and 4-bit Register

PS (address of the ROM) : 4-bits $\Rightarrow$ 16 rows in the ROM
NS' (content of the ROM) : 4-bits $\Rightarrow$ 4-bits: width of the ROM

---

Problem #2

a) 20-input decoder (using 4-input decoders)

$20/4 = 5 \Rightarrow 5$ levels
The outputs of each level is connected to the enable of next level decoders.

$\#$ decoders $= 1 + 16 + 16^2 + 16^3 + 16^4$
#2 Continued

b) 256-input mux (using 4-input mux) → \(256 = 2^8 \rightarrow (S_7 \ldots S_0)\)

\[
\begin{align*}
\frac{256}{4} &= 64 \\
\frac{64}{4} &= 16 \\
\frac{16}{4} &= 4 \\
\frac{4}{4} &= 1 \\
64 + 16 + 4 + 1 &\quad \text{max's} \quad 4 \text{ levels}
\end{align*}
\]

c) 32-bit Comparator (using 4-bit modules)

\[
\frac{32}{4} = 8 \\
\frac{8}{4} = 2 \\
8 + 2 + 1 = 11 \quad \text{comparator}
\]

d) 32-bit 3-shifter (using 8-bit 3-shifter) \(\frac{32}{8} = 4\)
# 2-d) continued

8-bit 3-shifter:

**Inputs:**

\[ X = (x_{10}, x_9, \ldots, x_0, x_z, x_{-z}, x_{-3}) \]

\[ S \in \{0, 1, 2, 3\} \]

\[ d \in \{L, R\} \]

\[ E \in \{0, 1\} \]

**Outputs:** \[ Y = (y_z, \ldots, y_0, y_1) \]

\[ x_i, y_i \in \{0, 1\} \]

\[ y_1 = \begin{cases} x_i - S & \text{if } d = L \text{ and } E = 1 \\ x_i + S & \text{if } d = R \text{ and } E = 1 \\ 0 & \text{if } E = 0 \end{cases} \]

**Assume:**

\[ L = 1 \Rightarrow d = 1 \]

\[ R = 0 \Rightarrow d = 0 \]

\[ s \]

\[ c_2 \]

\[ c_1 \]

\[ c_0 \]

\[ \rightarrow \text{ 8-bit 3-shifter: } 8 \text{ 8-input MUX's (one MUX per out} \]
Problem #3

ALU (add/sub/mul/div)

→ Assume we have a signal "start" which is "1" only in the beginning of each operation and "0" afterwards.

Notes: for mul: \[ S = S + a \]
For div: \[ S = S - b \]

For \( n \)-bits, \( n \)-bit output range = \([0, 2^n - 1]\)

Problem #4

(what you’ve learned in this course)

In this course we've learned to analyze and design both combinational and sequential systems. For combinational systems, we learned how to find truth table from the specification, and the logic function from the truth table, as well as how to minimize the number of gates (Boolean Algebra and Kmap).

For the sequential systems, we learned how to find state diagram/tables from the specification and how to use different types of Flip Flops to implement them.

We also became familiar with standard combinational modules such as decoders, encoders, mux's, adders, PLA, PAL etc. as well as standard sequential modules such as shift registers, counters, PSA, etc.